IBM Tivoli Enterprise Console

Adapters Guide

Version 3.8

GC32-0668-01

IBM Tivoli Enterprise Console

Adapters Guide

Version 3.8

GC32-0668-01

Note
Before using this information and the product it supports, read the information in "Notices" on page 165.

First Edition (September 2002)

This edition applies to version 3, release 8, of IBM Tivoli Enterprise Console (product number 5698-TEC) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Preface

The *IBM® Tivoli Enterprise Console® Adapters Guide* provides detailed descriptions for the currently available IBM Tivoli® Enterprise Console adapters.

Who Should Read This Guide

This guide is for IBM Tivoli Enterprise Console administrators who configure event adapters and IBM Tivoli Enterprise Console gateways.

You should have prior knowledge of the following:

- UNIX® operating system
- Microsoft® Windows® 2000 or Windows NT® operating systems
- Tivoli Management Framework
- Adapter operating system

  For example, if you are using an OpenView adapter, you should be familiar with Hewlett-Packard OpenView.

What This Guide Contains

The *IBM Tivoli Enterprise Console Adapters Guide* contains the following sections:

- Chapter 1, “Understanding Adapters”

  Describes adapters, events, attributes, adapter architecture, and adapter files.
- The following chapters provide information about how to configure and use each adapter:
  - Chapter 2, “AS/400 Alert Adapter”
  - Chapter 3, “AS/400 Message Adapter”
  - Chapter 4, “NetWare Log File Adapter”
  - Chapter 5, “OpenView Adapter”
  - Chapter 6, “OS/2 Adapter”
  - Chapter 7, “SNMP Adapter”
  - Chapter 9, “UNIX Log File Adapter”
  - Chapter 10, “Windows Event Log Adapter”
  - Chapter 11, “Windows NT Event Log Adapter”
- Chapter 8, “IBM Tivoli Enterprise Console Gateways”

  Provides information about how to configure the IBM Tivoli Enterprise Console gateway.
- Appendix A, “Files Shipped with Adapters”

  Lists significant files shipped with and used by each adapter.
- Appendix B, “Format File Reference”

  Contains details about format files, including organization, syntax, and how to modify them.
- Appendix C, “Class Definition Statement File Reference”

  Contains details about class definition statement files, including organization, syntax, and how to modify them.

Publications

This section lists publications in the *IBM Tivoli Enterprise Console* library and any other related documents. It also describes how to access Tivoli publications online and how to make comments on Tivoli publications.

IBM Tivoli Enterprise Console Library

The following documents are available in the *IBM Tivoli Enterprise Console* library:

- *Tivoli Event Integration Facility User's Guide*, GC32-0691

  Discusses how to develop your own event adapters that are tailored to your network environment and your specific needs. Additionally, the guide describes how to filter events at the source.

- *IBM Tivoli Enterprise Console Installation Guide*, GC32-0823

  Discusses how to install, upgrade, and remove IBM Tivoli Enterprise Console components.

- *IBM Tivoli Enterprise Console Reference Manual*, GC32-0666

  Provides details about command-line commands applicable to using the IBM Tivoli Enterprise Console product, the predefined tasks shipped in the task library, and the environment variables available to tasks that execute with an event.

- *IBM Tivoli Enterprise Console Rule Builder's Guide*, GC32-0669

  Discusses how to develop rules and integrate them for event correlation and automated event management.

- *IBM Tivoli Enterprise Console User's Guide*, GC32-0667

  Discusses how to plan for and configure your event database environment and describes components, roles, and other information for using the IBM Tivoli Enterprise Console product.

Prerequisite Publications

To be able to use the information in this book effectively, you must have some prerequisite knowledge, which you can get from the following books:

- *Tivoli Management Framework Planning for Deployment Guide*, GC32-0393

  Introduces the Tivoli environment and provides detailed information about the desktop, managed nodes, administrators, policy regions, profiles, notices, tasks, and scheduling.

- *Tivoli Management Framework User's Guide*, GC31-8433

  Describes the concepts and procedures for using Tivoli Management Framework services. It provides instructions for performing tasks from the Tivoli desktop and from the command line.

- *Tivoli Management Framework Reference Manual*, SC31-8434

  Provides information about the command line interface for Tivoli Management Framework.

Related Publications

The *Tivoli Glossary* includes definitions for many of the technical terms related to Tivoli software. The *Tivoli Glossary* is available, in English only, at the following Web site:

http://www.tivoli.com/support/documents/glossary/termsm03.htm

Accessing Publications Online

Publications in the product libraries are included in PDF or HTML formats, or both, on the product CD. To access publications using a Web browser, open the infocenter.html file, which is located in the appropriate publications directory on the product CD.

When IBM publishes an updated version of one or more online or hardcopy publications, they are posted to the Tivoli Information Center. You can access updated publications in the Tivoli Information Center from the following Customer Support Web site:

http://www.tivoli.com/support/documents/

The Tivoli Information Center contains the most recent version of the books in the product library in PDF or HTML formats, or both. Translated documents are also available for some products.

Note: If you print PDF documents on other than letter-sized paper, select the **Fit to page** check box in the Adobe Acrobat Print dialog (which is available when you click **File —> Print**) to ensure that the full dimensions of a letter-sized page are printed on the paper that you are using.

Providing Feedback about Publications

If you have comments or suggestions about Tivoli products and documentation, send an e-mail to `pubs@tivoli.com` or complete the customer feedback survey at the following Web site:

http://www.tivoli.com/support/survey/

Contacting Customer Support

If you have a problem with any Tivoli product, you can contact IBM Customer Support for Tivoli products. See the *Tivoli Customer Support Handbook* at the following Web site:

http://www.tivoli.com/support/handbook/

The handbook provides information about how to contact Customer Support, depending on the severity of your problem, and the following information:

- Registration and eligibility
- Telephone numbers and e-mail addresses, depending on the country in which you are located
- What information you should gather before contacting Customer Support

Conventions Used in this Guide

This book uses several conventions for special terms, actions, operating system-dependent commands, and paths.

Typeface Conventions

The following typeface conventions are used in this book:

Bold Commands, keywords, file names, authorization roles, URLs, or

	other information that you must use literally appear in **bold**. Names of windows, dialogs, and other controls also appear in **bold**.
Italics	Variables and values that you must provide appear in *italics*. Words and phrases that are emphasized also appear in *italics*.
Monospace	Code examples, output, and system messages appear in a `monospace` font.

Operating System-dependent Variables and Paths

This book uses the UNIX convention for specifying environment variables and for directory notation.

When using the Windows command line, replace **$***variable* with **%***variable***%** for environment variables and replace each forward slash (/) with a backslash (\) in directory paths.

Note: If you are using the bash shell on a Windows system, you can use the UNIX conventions.

Chapter 1. Understanding Adapters

Event adapters are software programs that collect information, perform local filtering, and convert relevant events into a format that can be used by the IBM Tivoli Enterprise Console product. Because adapters are located on or near their event sources and can perform local filtering of events, the adapters create a minimal amount of additional network traffic. Adapters use a minimal amount of system resources to perform their functions.

Network management applications have become an important part of monitoring the availability of resources in the enterprise. The IBM Tivoli Enterprise Console product can seamlessly integrate alarms and events from all the major network management platforms and can correlate them with other system, database, and application events.

Adapters are passive collectors of all types of events from systems and applications, including the network management applications. All of your existing network management configuration and monitoring of events can be preserved; these events can simply be forwarded to the event server for correlation with other events, where automated responses can be triggered or Information Technology (IT) staff can be notified.

Adapter Overview

An *adapter* is a process that monitors resources so that they can be managed. These monitored resources are called sources. A *source* is an application (for example, a database) or system resource (for example, an NFS server). When an adapter detects an event generated from a source (generally called a raw event), it formats the event and sends it to the event server. The event server then further processes the event.

Adapters can monitor sources in the following ways:

- An adapter can receive events from any source that actively produces them. For example, SNMP adapters can receive traps sent by the Simple Network Management Protocol (SNMP).
- An adapter can check an ASCII log file for raw events at configurable intervals if the source updates a log file with messages.

How Events Get Sent to the Event Server

Adapters can send events to the event server using a TME® interface or a non-TME interface. Both types of interfaces send events using an ordinary TCP/IP channel. The difference between the two interfaces is the method used to establish the connection. A *TME interface* establishes a connection using the **oserv** services provided by Tivoli Management Framework; therefore, adapters that use this interface are referred to as TME adapters. A *non-TME interface* establishes connections using standard interprocess communication mechanisms (for example, opening an IP socket); therefore, adapters that use this interface are called non-TME adapters.

How Events Get to the Event Server From an Endpoint

TME adapters installed on endpoints send their events to the **lcfd** process, which then sends the events to an IBM Tivoli Enterprise Console gateway, which in turn

bundles them up and forwards them on to an event server. A TME interface is used for communications. The IBM Tivoli Enterprise Console gateway uses a ***connection-oriented*** service to the server by default. A connection-oriented service means that a connection is established when the adapter is initialized and the connection is maintained for all events to be sent. The IBM Tivoli Enterprise Console gateway runs on the same managed node as the Tivoli Management Framework gateway that is providing the endpoint gateway service. The IBM Tivoli Enterprise Console gateway provides the following benefits:

- Greater scalability, meaning you can manage many sources easier, with less software running on the endpoints.
- Greatly reduces the amount of communications tasks performed by the event server or the Tivoli management region server, as the IBM Tivoli Enterprise Console gateway bundles a number of events before sending them to the event server. This improves event server performance.
- Easier deployment of adapters and updates to adapters using profiles in the Adapter Configuration Facility (ACF).

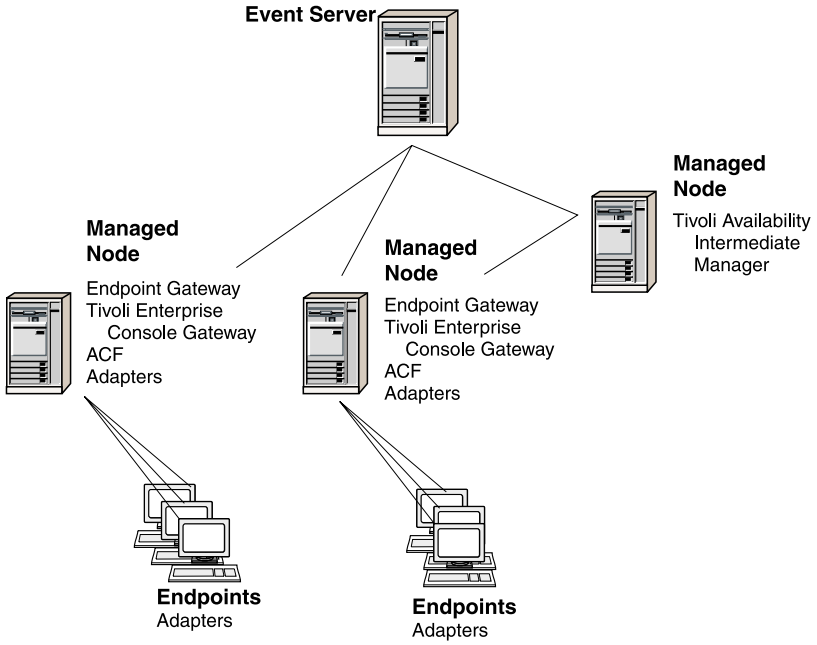
The TME adapters currently supported for an endpoint are the following:

- UNIX log file
- OS/2®
- SNMP
- Microsoft Windows event log
- Windows NT event log

You configure these adapters to send their events to specific primary, secondary or both event servers, and the IBM Tivoli Enterprise Console gateway forwards them appropriately. If the IBM Tivoli Enterprise Console gateway, Tivoli Management Framework gateway, or **lcfd** process is down, events are buffered at the endpoint. The events are re-sent when communication is restored and the next event is sent. If an event server is down (but the IBM Tivoli Enterprise Console gateway, Tivoli Management Framework gateway, and **lcfd** processes are still up), events are buffered at the IBM Tivoli Enterprise Console gateway. They are re-sent when communication with the server is restored and the next event is sent.

The IBM Tivoli Enterprise Console gateway has configuration options that can be specified similarly to how configuration options are specified for an adapter; that is, you can configure the IBM Tivoli Enterprise Console gateway with a configuration file that you distribute to the gateway node endpoint. For details about configuring an IBM Tivoli Enterprise Console gateway, see Chapter 8, “IBM Tivoli Enterprise Console Gateways” on page 95.

The following figure shows an example of the IBM Tivoli Enterprise Console product and Tivoli Management Framework component relationships in a network with endpoints.



How Events Get to the Event Server From a Managed Node

For network management OpenView adapters, events are sent from the managed node adapter directly to the event server using a TME interface. In other words, the **oserv** of the managed node that the adapter runs on sends the event to the **oserv** of the event server when these are separate nodes, which then forwards it on to the event server process.

For the UNIX log file, OS/2, Windows, Windows NT, and SNMP TME adapters, a managed node must also be configured as an endpoint to send events to the event server.

How Events Get to the Event Server From a Non-TME Adapter

A non-TME adapter sends events directly to the event server using an IP socket.

Internationalization Support for Events

By default, the following log file adapters send their events to the event server in UTF-8 encoding:

- UNIX log file adapter
- NetWare log file adapter
- OS/2 log file adapter
- Windows event log adapter
- Windows NT event log adapter

To change the default configuration of these adapters so they send events in the encoding of the event server host instead of UTF-8, the **Pre37Server** and **Pre37ServerEncoding** configuration file options are provided. See page 12 for additional information about these options.

The event server can receive events in both UTF-8 encoding or the encoding of the event server host. The event server automatically determines the type of encoding (UTF-8 or non-UTF-8) of an event by evaluating a particular flag in the event data.

The adapter automatically reads the format file from the appropriate directory. If the adapter is sending events to an event server running a version earlier than the IBM Tivoli Enterprise Console 3.7 product, the format files in the localization directories must remain in English. See "Format File" on page 17 and Appendix B, "Format File Reference" on page 145 for additional information.

Tivoli Event Integration Facility provides support for creating new adapters (other than those shipped by the IBM Tivoli Enterprise Console product) or modifying existing adapters to send events to the latest version of the event server. Existing adapters shipped in a previous release of the IBM Tivoli Enterprise Console product do not require updating; the new event server recognizes events sent from those adapters. See the *Tivoli Event Integration Facility User's Guide* for additional information.

When the adapter is installed, a new **codesets** directory appears with the **bin** and **etc** directories under **$TECADHOME**.

Event Information

Event information is formatted as a set of attributes. Each attribute is predefined and contains a name and value. Adapters separate information into event classes, format this information into attributes, and send this information to the event server. The event server then processes this information.

Event classes are a classification of events; do not confuse them with the term classes in the traditional object-oriented sense. Event classes can be subclassed to facilitate a further breakdown of information so that more detailed rules can be applied to the information. In essence, event classes are an agreement between the adapter and the event server about what information the adapter sends to the event server for a given class.

After event information is separated into attributes and the event is categorized into an event class, the adapter sends the information to the event server for further processing. Adapters are configured to send only information that administrators are interested in; that is, filters are established on the local system that specify whether to discard an event or forward it to the event server. This minimizes any network loading that is related to enterprise monitoring.

Event Attributes

An event class name is followed by attribute information.

An adapter supplies information in the form of attributes. An attribute has the following format:

```
attribute_name=value
```

The following list describes base event attributes that can be contained in an event sent to the event server. Base event attributes are standard for most event classes and are defined in the highest superclass of a basic recorder of objects in C (BAROC) file. An adapter can also contain adapter-specific or user-defined attributes.

Attribute Name	Contents
acl	The list of authorization roles that enables an administrator to modify the event.
adapter_host	The host on which the adapter is running.
administrator	The administrator who acknowledged or closed the event.
cause_date_ reception	The **cause_date_reception** attribute is used to link an effect event to its cause event. This value is set to the value of the **date_reception** attribute of the cause event.
cause_event_ handle	The **cause_event_handle** attribute is used to link an effect event to its cause event. This value is set to the value of the **event_handle** attribute of the cause event.
credibility	Indicates how the event was sent from the adapter. The value is 1 if an event was sent using a communications channel provided by Tivoli Management Framework services, as is the case for a TME adapter. The value is zero (0) if an event was sent from a non-TME adapter.
date	The date and time the event was generated.
date_reception	A time stamp indicating the time the event server received the event. It is an integer representing the number of seconds since the epoch, which is January 1, 1970. This value is also used as a component to uniquely identify an event. An event is uniquely identified by a combination of the values for the **date_reception**, **event_handle**, and **server_handle** attributes.
duration	For closed events, the age (in seconds) of the event from when it was received by the event server until it was closed. For all non-closed events, the value is zero (0). **Note:** If an event was closed by calling the **set_event_status** predicate from within a rule, this attribute is not modified to give the age. The value remains at zero (0).
event_handle	A number used to reference the event. An event is uniquely identified by a combination of the values of the **date_reception**, **event_handle**, and **server_handle** attributes. Events received within the same second are assigned an incremental number for this attribute starting at 1 and incremented by 1.
hostname	The name of the system on which the event occurred.
msg	A text summary of the event.
msg_catalog	For future support of internationalized event messages; not currently implemented.
msg_index	The message ID used to obtain the internationalized message.
num_actions	The number of actions (tasks or programs) currently being tracked by the event server for this event.
origin	The protocol address or host name of the source system.
repeat_count	A counter for keeping track of the number of times a duplicate type of event has been received.
server_handle	A number identifying the event server that received this event. An event is uniquely identified by a combination of the values for the **date_reception**, **event_handle**, and **server_handle** attributes.

Attribute Name	Contents

server_path	Stores information describing the rule engines that an event has passed through. **server_path** has the following definition:
	`server_path list_of_strings;`
	Each element in the list represents one rule engine that the event has visited, and each element contains a rule engine identifier, server number, reception ID, and event handle. The following is an example of a list:
	`chair 1 12121212 3`
	where:
	chair The rule engine identifier
	1 The server number
	12121212 The event reception ID in server 1
	3 The event handle for the event in server 1

severity	The severity of the event. The database stores the severity as a number. This mapping is defined in the **root.baroc** rule base file and is set for the event server default severities as follows: **10** UNKNOWN **20** HARMLESS **30** WARNING **40** MINOR **50** CRITICAL **60** FATAL You can also customize the severity settings.
source	The source of the event (for example, the OpenView adapter). The source is defined by the adapter type.

Attribute Name	Contents
status	The status of an event. It is initially set to OPEN or to a default value specified by the event class. Possible values during an event lifetime are as follows: **ACK** An administrator or rule has acknowledged the event. **CLOSED** An administrator or rule has fixed the problem that was reported by the event. An event adapter can also send an event with a status of **CLOSED** to indicate that a previously received event of the specified class should have its status changed to **CLOSED**; the previously received event to be closed is the most recent duplicate of the same event. The event being sent with a **CLOSED** status is dropped and not stored in the event database. *custom_status* A status that has been added to the **STATUS** enumeration for site-specific purposes. The **STATUS** enumeration is defined in the **root.baroc** file. To add a new status, edit this file, recompile the rule base, and restart the event server. **OPEN** The event has been received by the event server, but no administrator or rule has acknowledged it. **RESPONSE** A rule has automatically responded to the event. This status is assigned a rule language predicate. It is not available from an event console. The database stores the status as a number. This mapping is defined in the **root.baroc** rule base file and is set for the event server default status as follows: zero (0) for OPEN, 10 for RESPONSE, 20 for ACK, 30 for CLOSED.
sub_origin	A further categorization of the origin. This attribute is optional.
sub_source	A further categorization of the source. This attribute is optional.

The adapter uses the following attributes to uniquely identify an event:

- **date_reception**
- **event_handle**
- **server_handle**

Adapter Files

An adapter uses various files for its operations. The following table provides a brief description of the types of files that can be used. Subsequent sections discuss some of the more common files you might need to view or modify for configuration or troubleshooting purposes. See Appendix A, “Files Shipped with Adapters” on page 141 for detailed information about which files are shipped with particular adapters.

File Type	Description
Basic recorder of objects in C (BAROC)	Defines event classes to the event server; must be part of the rule base.
Cache	Stores buffered events.
Class definition statement (CDS)	Defines event class definitions to the adapter.

File Type	Description
Configuration	Defines configuration options for adapters.
Error	Defines error logging and tracing options for the adapter.
Format	Defines the format of messages and matches them to event classes for the UNIX log file, NetWare log file, OS/2, and Windows and Windows NT event log adapters.
Installation script	Configures the adapter to start when the operating system starts.
Object identifier	Defines object-identifier-to-name mappings for the NetView® /6000, OpenView, and SNMP adapters.
Registration	The registration file generated by the installation script for NetView/6000 and OpenView.
Rules	Defines rules to the event server; must be part of the rule base.

An adapter uses the **TIVOLI_COMM_DIR** Tivoli Management Framework environment variable, if set, to determine which directory to use for its lock and pipe files. If the variable is not set, **/tmp/.tivoli** is used instead. For more information about this environment variable, see the *Tivoli Management Framework Release Notes*.

Cache File

Events are written to the cache file using a "circular" method; when the cache file has reached the size limit set by **BufEvtMaxSize**, the next new event is written to the beginning of the cache file (thus overwriting the existing data at that location). Subsequent events continue being written in order until the end of the file is reached again, and the process starts over from the beginning of the file. A small header at the beginning of the file tracks where the next new event will be written and where the next old event will be removed.

The format of the cache file is as follows:

```
 Cache File Format:
-----------------
maxsz: XXXXXXXXXX
head : XXXXXXXXXX
tail : XXXXXXXXXX
.......................event1 event2
event3 event4 event5.................
.....................................
.....................................
.....................................
```

The first three lines in the cache file all have a fixed size of 18 bytes and contain the following data:

maxsz The maximum size of the cache file.

head The byte offset from the beginning of the file to the next event to send. A value of zero (0) indicates an empty cache file.

tail The byte offset from the beginning of the file to the first byte of free space in the file.

The boundaries between events in the cache file are indicated by a terminating **^A** character at the end of each event.

Configuration File

Most adapters come with a configuration file containing configuration options and filters. This file is read by an adapter when it is started. By modifying this file, you can reconfigure an adapter at anytime, without having to modify the adapter source code. To have your configuration changes take effect, simply stop and restart the adapter. A configuration file usually has an extension of **.conf**; see each specific adapter chapter for exact file names.

File Location

By default, an adapter expects its configuration file (along with its format, CDS, and error files) to be located as shown in the following table. For Windows and Windows NT, the syntax shown is correct when running the bash interpreter.

Adapter Type	Node Type	Location
TME	Managed node	**$BINDIR/TME/TEC/adapters/etc/** or **/etc/Tivoli/tecad/etc** (which is a link to the TME adapter directory)
	Endpoint	**$LCFROOT/bin/$INTERP/TME/TEC/adapters/etc** or **/etc/Tivoli/tecad/etc** (which is a link to the TME adapter directory)
non-TME	Not applicable	*path***/etc** where the adapter was manually installed or **/etc/Tivoli/tecad/etc** (which is a link to the TME adapter directory)

For information about directory structures and system variables (those beginning with $), see the *Tivoli Management Framework Planning for Deployment Guide.*

File Format

Each non-blank line that does not begin with the comment sign (#) is of one of the following forms:

- To specify configuration options:

  keyword=value

- To specify event filters:

  `Filter:CLASS=class_name;attribute=value;`

- To specify event buffer filters:

  `FilterCache:CLASS=`*`class_name`*`;`*`attribute=value`*`;`

Example

```
#
# Communication Parameters
#
ServerLocation=ravel
ServerPort=5529
#
# Event Filters
#
Filter:Class=disk_event
Filter:Class=Su_Success;origin=126.32.2.14
```

Keywords

Keywords use the following format: *keyword=value*

Some adapters have additional keywords specific to them. See each specific adapter chapter for descriptions of these keywords. Adapters do not issue error messages for misspelled keywords or keywords set to a value that is not valid. Do not use blank spaces in keyword statements unless enclosed in single quotation marks (however, you cannot use quotation marks at all with the **HPOVFilter** keyword in the HPOV adapter). Do not use class names not defined in a BAROC file with configuration options.

A configuration file can contain the following keywords, which are common to most adapters:

AdapterCdsFile=*path*
: Specifies the full path name of the CDS file. This keyword is required if the CDS file is not in the same directory as the configuration file.

AdapterErrorFile=*path*
: Specifies the full path name of the error file. This keyword is required if the error file is not in the same directory as the configuration file.

BufEvtMaxSize
: Specifies the maximum size, in kilobytes, of the adapter cache file. The default value is 64. The cache file stores events on disk when they cannot be sent to the event server.

  The **BufEvtMaxSize** keyword is optional.

BufEvtPath
: Specifies the full path name of the adapter cache file. On endpoint adapters, the **BufEvtPath** keyword uses the **$TIVOLIHOME** variable to resolve file location and drive letter differences over different environments by using a path relative to the endpoint installation. The ACF defines **$TIVOLIHOME** on each endpoint; you cannot change its value.

Operating System	Default Path	$TIVOLIHOME Value
UNIX	**$TIVOLIHOME/tec/ tecad_***adapter***.cache**	**/etc/Tivoli**
Windows, Windows NT	**$TIVOLIHOME\tec\ tecad_***adapter***.cache**	**%SystemRoot%\system32\ drivers\etc\Tivoli**

  The AS/400® adapters do not use this keyword.

  This keyword is required when the **BufferEvents** keyword is set to YES.

BufferEvents
: Specifies whether or not event caching is enabled. If **BufferEvents** is set to anything other than **YES**, events are not cached. The value is not case-sensitive. The default value is **YES**.

  The **BufferEvents** keyword is optional.

BufferFlushRate
: Specifies the number of events sent per minute. Once the adapter has recovered the lost connection, and there are events in the buffer, the events are sent at this rate per minute. The default value is zero (0); all events are sent in one burst.

  The **BufferFlushRate** keyword is optional.

ConnectionMode
: Specifies the connection mode to use to connect to the IBM Tivoli Enterprise Console gateway or event server. Valid values are

connection_oriented (or its abbreviations **CO** and **co**) and **connection_less**. The default value is **connection_less**, except for the AS/400 adapters and the IBM Tivoli Enterprise Console gateway, which have **connection_oriented** as the default value.

When **connection_less** is specified or used by default, a new connection is established (and discarded) for each event or group of events that is sent. When **connection_oriented** or one of its abbreviations is specified, a connection is established at adapter initialization and is maintained for all events sent. A new connection is established only if the initial connection is lost. The connection is discarded when the adapter is stopped.

The **ConnectionMode** keyword is optional.

Filter Works with the **FilterMode** keyword to determine how events are filtered. An event matches a **Filter** statement when each *attribute=value* pair in the **Filter** statement is identical to the corresponding *attribute=value* pair in the event.

A **Filter** statement must contain the event class, and optionally can include any other *attribute=value* pair that is defined for the event class. The format of a filtering statement is the following:

```
Filter:Class=class_name;[attribute=value;...;attribute=value]
```

Each statement must be on a single line. The *attribute=value* pair is case sensitive.

This keyword is optional.

FilterCache

Works with the **FilterMode** and **Filter** keywords to determine which events are stored in the cache when events cannot be sent successfully to the event server. To store events in the cache, you must set **BufferEvents**=YES. An event matches a **FilterCache** statement when each *attribute=value* pair in the **FilterCache** statement is identical to the corresponding *attribute=value* pair in the event.

A **FilterCache** statement must contain the event class (*class_name*) and can include any *attribute=value* pair that is defined for that event class. The format of a filtering statement is the following:

```
Filter:Class=class_name;[attribute=value;...;attribute=value]
```

Each statement must be on a single line. The *attribute=value* pair is case sensitive. You must specify the **Filter** keyword, when you use the **FilterCache** keyword. Additionally, the **FilterCache** statement must specify the same class or subset of classes that the **Filter** statement specifies.

This keyword is optional.

Note: When using **FilterCache** with endpoint adapters and the IBM Tivoli Enterprise Console gateway, you must set the filtering statements at both locations to the same specifications.

FilterMode

Specifies whether events that match a **Filter** or **FilterCache** statement are sent to the event server (**FilterMode**=IN) or discarded (**FilterMode**=OUT). The default value is OUT. The valid values are IN or OUT, without regard for case. If you set **FilterMode**=IN, you must have one or more **Filter** and **FilterCache** statements defined.

For information about how to use filtering keywords to send, cache, and discard events, see “Event Filtering” on page 14.

This keyword is optional.

getport_timeout_seconds
Specifies the number of seconds to wait before re-sending the UDP call for a port, *if* no response is heard. It re-transmits until the RPC call times out. The default value is zero (0) seconds.

getport_timeout_usec
Specifies the number of microseconds to add to the seconds specified with the **getport_timeout_seconds** keyword. The default value is 50 000 microseconds.

getport_total_timeout_seconds
Specifies the number of seconds to wait on getting a port after making a all to the portmapper. The default value is zero (0) seconds.

getport_total_timeout_usec
Specifies the number of microseconds to add to the seconds specified with the **getport_total_timeout_seconds** keyword. The default value is 50 000 microseconds.

NO_UTF8_CONVERSION
Specifies whether to encode event data in UTF-8. When this options is set to **YES**, the IBM Tivoli Enterprise Console product does not encode event data in UTF-8. The data is assumed to already be in UTF-8 encoding when passed to the IBM Tivoli Enterprise Console product. It does, however, prepend the flag indicating that the data is in UTF-8 encoding if the flag does not exist at the beginning of the event data.

The default value for this option is **NO**.

Pre37Server

Specifies whether the adapter is to send its events in the encoding of the event server host or in UTF-8 encoding. Event server host versions earlier than the IBM Tivoli Enterprise Console 3.7 product do not support UTF-8 encoding of events. When set to **YES**, this keyword disables UTF-8 encoding and allows the adapter to communicate with event server host versions earlier than the IBM Tivoli Enterprise Console 3.7 product. When this keyword is set to **NO**, the adapter sends events in UTF-8 encoding. The values are not case-sensitive. The default is **NO**.

When this keyword is set to **YES**, you must also specify the **Pre37ServerEncoding** keyword.

Pre37ServerEncoding

Determines which language to use when a non-TME adapter communicates with a non-UTF-8 event server host (versions earlier than the IBM Tivoli Enterprise Console 3.7 product). This keyword is active only when **Pre37Server** is set to **YES**. This keyword only applies to the log file adapters (UNIX, NetWare, OS/2, Windows, and Windows NT).

RetryInterval

When **ConnectionMode=connection_oriented**, and the connection to the event server is lost, an adapter waits the specified number of seconds before connecting to a secondary server or buffering the events. While the adapter is waiting for the expiration of this interval, no new events are processed by the adapter.

This option allows an adapter to send all events to the primary event server even if the primary event server is stopped briefly, such as when loading a new rule base.

If you use this option to wait for restarting an event server, set the value for a period of time longer than necessary for the event server to be stopped and then restarted.

The **RetryInterval** keyword is optional. The default is 120 seconds.

ServerLocation

Specifies the name of the host on which the event server is installed. The value of this field must be one of the formats shown in the following table, depending on whether the adapter is a TME adapter or a non-TME adapter, and whether the event server is part of an interconnected Tivoli management region:

Adapter Type	Format
TME	**EventServer**
TME in an interconnected Tivoli management region	**EventServer**#*region_name*
non-TME	*host_name* or *IP_address*. Use the dotted format for *IP_address*.

Note: AS/400 adapters are non-TME adapters.

For TME adapters on managed nodes and non-TME adapters, **ServerLocation** can contain up to eight values, separated by commas. The first location is the primary event server, while others are secondary servers to be used in the order specified when the primary server is down.

For endpoint adapters, secondary event servers, if any, are defined in the IBM Tivoli Enterprise Console gateway configuration file. Only specify a primary event server in an endpoint adapter configuration file.

The default is **EventServer**. To use a non-TME value for **ServerLocation**, see "Configuration File" on page 97 for more information.

The **ServerLocation** keyword is required.

Note: **ServerLocation** defines the path and name of the file for logging events, instead of the event server, when used with the **TestMode** keyword.

ServerPort

Specifies the port number on a non-TME adapter on which the event server listens for events. Set this keyword value to zero (0), the default value unless the portmapper is not available on the event server, which is the case if the event server is running on Windows or the event server is a Tivoli Availability Intermediate Manager (see the following note). If the port number is specified as zero (0) or it is not specified, the port number is retrieved using the portmapper.

The **ServerPort** keyword can contain up to eight values, separated by commas. For non-TME adapters that send events to a UNIX event server, use the default value of zero (0) (only one value of zero, even if multiple UNIX event servers are specified with the **ServerLocation** keyword). For

non-TME adapters that send events to a Windows event server or a Tivoli Availability Intermediate Manager (AIM), specify one value for each event server defined with the **ServerLocation** keyword.

The **ServerPort** keyword is optional when the event server is running on UNIX, but mandatory when running on Windows.

Note: *If the event server is running on Windows:* There is no portmapper daemon on a Windows machine that allows the adapter to query the reception port at runtime. The event server listens on a fixed reception port (**tec_recv_agent_port** in **.tec_config**) for connection and adapter input. Set **ServerPort** to the value of the **tec_recv_agent_port** entry in the **.tec_config** file in the **$BINDIR/TME/TEC** directory. The default is 5529. The Tivoli Availability Intermediate Manager never uses the portmapper; the Tivoli Availability Intermediate Manager server listens on a fixed port set in the Tivoli Availability Intermediate Manager graphical user interface.

TestMode

Specifies whether test mode is turned on or off. When **TestMode=YES**, the **ServerLocation** keyword specifies the file to which events are logged, instead of being sent to the event server. Valid values are **YES** and **NO**, without regard to case. The default is **NO**.

The **TestMode** keyword is optional.

Event Filtering

Normally, an adapter sends all events to the event server. You can optionally specify events that can or cannot be sent to the event server. You can do this by specifying the event class and such information as the origin, severity, or any other *attribute=value* pair that is defined for the event class. The class name specified for an event filter entry must match a defined class name; an adapter does not necessarily have knowledge of the class hierarchy.

Depending on how you specify the **Filter** and **FilterMode** keywords, filtered events are either sent to the event server or discarded.

- To send specific events to the event server:
  1. Set **FilterMode** to IN.
  2. Create **Filter** statements to match the specific events that you want sent.
- To discard specific events:
  1. Set **FilterMode** to OUT (the default value).
  2. Create **Filter** statements to match the specific events that you want discarded.
- To send all events to the event server (the default behavior):
  1. Set **FilterMode** to OUT.
  2. Do *not* specify any **Filter** statements.

Note: All events are discarded when the configuration is as follows:

1. **FilterMode** is set to IN.
2. No **Filter** statements are specified.

To use non-English characters in a **Filter** statement, you must enter the non-English characters in the local encodings.

Regular Expressions in Filters: You can also use Tcl regular expressions in filtering statements. The format of a regular expression is `re:'value_fragment'`.

Note: Tivoli Event Integration Facility uses an exception to the Tcl regular expression syntax. The backslash character (\) in Tivoli Event Integration Facility indicates that the following literal character is the character to filter for, not some special character such as a tab. For example, `\t` means the tab character in Tcl, but means `t` in Tivoli Event Integration Facility.

The following example shows a **Filter** statement with a regular expression. This filter statement matches all events with a class name that contains `TEC_` somewhere in its name:

```
Filter:Class=re:'TEC_.*'
```

The following example shows a **FilterCache** statement with a narrower range. This filter statement matches all events with a class name that contains `TEC_` somewhere in its name *and* has a severity of critical:

```
FilterCache:Class=re:'TEC_.*';severity=CRITICAL
```

For more information about Tcl regular expressions, see a Tcl user's guide.

Event Filter Examples: The following table shows some event filter examples for a few different adapters:

Adapter	Example
AS/400 Alert	The following entry matches all events of the **SNA_Equipment_Malfunction** class from the origin 1.2.3.4: `Filter:Class=SNA_Equipment_Malfunction;origin=1.2.3.4`
UNIX Log File	The following entry matches all events of the **Su_Success** class from the origin 126.32.2.14: `Filter:Class=Su_Success;origin=126.32.2.14`
OpenView	The following entry matches all events of the **OV_Message** class from the origin 126.32.2.14: `Filter:Class=OV_Message;origin=126.32.2.14`
Windows NT	The following entry matches all events of the **NT_Power_Failure** class from the origin 126.32.2.14: `Filter:Class=NT_Power_Failure;origin=126.32.2.14`

Event Buffer Filtering

When an adapter is unable to connect to the event server or IBM Tivoli Enterprise Console gateway, it sends the events to a file if the **BufferEvents** keyword is set to `YES`. You can filter events sent to a cache file, similar to filtering events for the event server by using the **FilterCache** keyword.

There are no default event cache filters in the configuration files shipped with adapters.

The following procedures describe how to filter events with the **FilterCache** and **FilterMode** keywords, when the event server is unavailable:

- To cache specific events:
  1. Set **FilterMode** to IN.
  2. Set **BufferEvents** to YES (the default value).

3. Create **Filter** and **FilterCache** statements to match the specific events that you want cached.

- To discard specific events:
  1. Set **FilterMode** to OUT.
  2. Create **Filter** and **FilterCache** statements to match the specific events that you want discarded.
- To cache all events (the default behavior):
  1. Set **FilterMode** to OUT.
  2. Set **BufferEvents** to YES.
  3. Do *not* specify any **FilterCache** statements.

Note: All events are discarded when the configuration is as follows:

1. **FilterMode** is set to IN.
2. No **FilterCache** statements are specified.

Event Buffer Filter Examples: The following table shows some event buffer filter examples for a few different adapters:

Adapter	Example
AS/400 Alert	The following entry matches all events of the **SNA_Equipment_Malfunction** class from the origin 1.2.3.4: `FilterCache:Class=SNA_Equipment_Malfunction;origin=1.2.3.4`
UNIX Log File	The following entry matches all events of the **Su_Success** class from the origin 126.32.2.14: `FilterCache:Class=Su_Success;origin=126.32.2.14`
OpenView	The following entry matches all events of the **OV_Message** class from the origin 126.32.2.14: `FilterCache:Class=OV_Message;origin=126.32.2.14`
Windows NT	The following entry matches all events of the **NT_Power_Failure** class from the origin 126.32.2.14: `FilterCache:Class=NT_Power_Failure;origin=126.32.2.14`

BAROC File

Each adapter comes with a BAROC file describing the classes of events the adapter supports. This file is not used by the adapter itself, but serves as a mandatory link between the adapter and the event server. The event server must load this file before it is able to understand events received from the adapter. A BAROC file has an extension of **.baroc**; see each specific adapter chapter for exact file names. The format of a BAROC file is described in the *IBM Tivoli Enterprise Console Rule Builder's Guide*.

Example

The following fragment shows how an event class for reporting SNMP authentication problems could be defined in a BAROC file:

```
CLASS AUTHENTICATION_FAILURE ISA EVENT
DEFINES {
  source:default="SNMP";
  sub_source:default="NET";
  auth_source:STRING;
  };
END
```

Rule File

Some adapters come with a rule file describing the classes of events the adapter supports. This file is not used by the adapter itself, but serves as a mandatory link between the adapter and the event server. The event server must load this file before it is able to understand events received from the adapter. A rule file has an extension of **.rls**; see each specific adapter chapter for exact file names. The format of a rule file is described in the *IBM Tivoli Enterprise Console Rule Builder's Guide*.

Example

The following fragment shows how an event class for reporting SNMP authentication problems could be defined in a BAROC file:

```
CLASS AUTHENTICATION_FAILURE ISA EVENT
DEFINES {
  source:default="NET";
  sub_source:default="SNMP";
  auth_source:STRING;
  };
END
```

Format File

The UNIX log file, NetWare log file, OS/2, Windows, and Windows NT event log adapters can extract information from system log messages, whose format and meaning can vary widely. This capability is necessary because similar sources can produce messages in different formats. For example, different NFS (network file system) implementations might report the **file system full** error in different formats. As a result, you might need to match different messages to the same or different event classes. This type of matching is done with a format file.

The purposes of a format file are as follows:

- Serves as the lookup file for matching messages to event classes. When the format file is being used for this purpose, all format specifications in the file are compared from top to bottom. In situations where there are multiple matching classes for a message, the last matching format specification is used. If no match is found, the event is discarded.
- Serves as the source from which a CDS file is generated. See "Class Definition Statement File" on page 18 for additional information.

See Appendix B, "Format File Reference" on page 145 for details about format files.

Example

The following examples show sample entries from the format file used by the Windows NT event log adapter.

Note: The format files for the log file-type adapters are examples only; customization might be required. The message text must fit on one line and be no longer than 1024 characters.

```
FORMAT NT_Base
%t %s %s %s %s %s %s %s*
hostname DEFAULT
origin DEFAULT
category $3
eventType $4
sid $5
sub_source $6
id $7
msg $8
```

```
-date1 $1
-date2 $2
date PRINTF("%s %s", date1, date2)
END

FORMAT NT_Share_Dir_Missing FOLLOWS NT_Base
%t %s %s %s %s %s %s The server service was unable to recreate
the share %s because the directory %s no longer exists.
sharename $8
directoryname $9
END

FORMAT NT_Service_Start FOLLOWS NT_Base
%t %s %s %s %s %s %s %s* started successfully.
service $8
END

FORMAT NT_Service_Started FOLLOWS NT_Base
%t %s %s %s %s %s %s The %s* service was started.
service $8
END
```

Class Definition Statement File

CDS files are used by an adapter to map incoming raw events to a particular class and to define event attributes before forwarding the event to the event server.

No alterations to this file are necessary to use an adapter unless you alter the corresponding **.fmt** file (if any). If any event definition is changed in a CDS file, the corresponding event class definition in the BAROC file might need changing as well. Event definition content and syntax are discussed in the *IBM Tivoli Enterprise Console Rule Builder's Guide*.

See Appendix C, "Class Definition Statement File Reference" on page 155 for details about CDS files.

Example

The following example shows a CDS file:

```
#
# Default attribute values
#
MAP_DEFAULT
  source = SNMP;
  sub_source = NET;
# forwarding_agent = $SOURCE_ADDR;
  origin = $AGENT_ADDR;
  adapter_host = $ADAPTER_HOST;
END

CLASS Authentication_Failure_Cisco
  SELECT
    1: ATTR(=,$ENTERPRISE), VALUE(PREFIX, "1.3.6.1.4.1.9");
    2: $TYPE = 4;
    3: ATTR(=,"authAddr");
  FETCH
    1: IPNAME($SOURCE_ADDR);
  MAP
    hostname = $F1;
    originating_address = $V3;
END
# For Cisco routers, because we know the interface generating the trap,
# we map 'linkUp' traps to 'linkDown' CLOSED events
CLASS Link_Down_Cisco
  SELECT
    1: ATTR(=,$ENTERPRISE), VALUE(PREFIX, "1.3.6.1.4.1.9");
    2: $TYPE = 3;
    3: ATTR(=,"ifIndex");
```

```
    4: ATTR(=,"ifDescr");
    5: ATTR(=,"ifType");
    6: ATTR(=,"locIfReason");
  FETCH
    1: IPNAME($SOURCE_ADDR);
  MAP
    hostname = $F1;
    sub_origin = $V4;
    status = CLOSED;
    interface_index = $V3;
    interface_description = $V4;
    interface_type = $V5;
    reason = $V6;
END
```

Error File

It is possible to selectively activate tracing for any module of an adapter (parser, kernel, select, fetch, map, driver, and so forth) and for any level of error tracing. A different log file can be specified for each module/level pair. To see a continuous flow of adapter processing with tracing, change all occurrences of **/dev/null** to the same output file. Keep in mind that these tracing features can consume large amounts of disk space.

Note: The AS/400 adapters run in batch as an AS/400 job. Every job writes messages (completion, error, and informational) to a job log. See the AS/400 adapter chapters for more information about debugging and tracing options.

Specifications in the error file allow you to configure tracing options for an adapter. An error file usually has an extension of **.err**; see each specific adapter chapter for exact file names. An error file is located in the same directory as the adapter configuration file (see "File Location" on page 9 for details).

Note: The error file name can be specified in the configuration file by the **AdapterErrorFile** keyword, as shown in the following example:

```
AdapterErrorFile=/usr/tecad/tecad_adaptername.err
```

If you change event definitions in the CDS or format files, you can use the error file to confirm that the adapter works properly with the new event definitions.

To specify the exact path of the trace file, change all instances of **/dev/null** in the error file a file name that you want.

Each line of the error file consists of the following information:

module_name error_level output_file

where:

module_name Specifies the type of function to trace. Valid values are the following:

ERROR
An error function.

UTILS
A utility function.

PARSER
A parsing function.

KERNEL
A general kernel operation.

SELECT
A selection process.

FETCH
A fetch process.

MAP A mapping process.

DRIVER
A driver main program.

DRVSPEC
An SNMP specific driver part.

TECIO
An event server I/O.

error_level Specifies the type of error to look for or the type of trace to perform. Valid values are the following:

MINOR
A minor error.

MAJOR
A major error (running continues).

FATAL
A fatal error (running ends).

LOW Minimal tracing.

NORMAL
Normal tracing.

VERBOSE
Verbose tracing.

output_file Specifies the name of the file to write output to.

Initial Files

Each adapter comes with an initial set of files that provides out-of-the-box support for a predefined set of events. The set of files is composed of the following files:

- BAROC file
- CDS file
- For the adapters on NetWare, OS/2, UNIX, Windows, and Windows NT: format file

By modifying these files, a system administrator can add, modify, and specialize classes of events.

The number of different events an adapter can receive is infinite. Therefore, the major objective of the initial files provided with an adapter is not to be exhaustive, but essentially to support the most common type of events handled by this adapter (for example, SNMP generic traps), as well as to provide enough examples to the system administrator on which to build new event definitions.

The initial supported events for the adapters are described in each adapter chapter later in this guide.

Troubleshooting Adapters

The following sections list troubleshooting guidelines for the different types of adapters.

Adapter Startup Errors

If the adapter fails to start, look in the **/tmp** directory for the **tecadEH.log** file. You might be able to learn why the adapter failed from reading this file. The following list shows examples of errors you might find in **tecadEH.log**:

```
tecad EH : error 2 invalid error config line: Normal
tecad EH : error 4 Init: Stat failed on error file </etc/tecad_hpov.err>
```

All Adapters

1. You receive a connection error when using **wpostemsg** or **postemsg**. The error indicates that you might be using a user ID other than Administrator or root. Thus, your ID does not have the correct permissions to create and write the file specified by the **BufEvtPath** keyword.
2. If the adapter receives the event and you can determine (through tracing or debugging) that the event matches the correct class, use the tracing output to verify if the event was sent to the event server, not sent, or cached. If the event was not sent to the event server, check the adapter configuration file to see if that class was filtered out.
3. If the event was sent to the event server, verify that the event server is actually running. Then run the **wtdumprl** command to check to see if the event server received the event but failed to parse the event correctly. Also check the current rule base rules to see if the event was dropped. See the *IBM Tivoli Enterprise Console Reference Manual* for more information about **wtdumprl**.
4. Check the cache files to see if the event was cached.

Managed Node Adapters

1. Use the tracing and debugging options detailed in each chapter. This helps determine if the adapter receives the event and how the adapter handles the event.
2. Use Tivoli Management Framework debugging output of the **odstat** and **wtrace** services. These services show what occurs after the adapter tries to send an event from the managed node **oserv** service to the IBM Tivoli Enterprise Console **oserv** services, and they also help debug problems that occur during Adapter Configuration Profile (ACP) distributions.
3. Use the managed node **wpostemsg** command from the system the adapter is running on to see if the event arrives at the event server. See the *IBM Tivoli Enterprise Console Reference Manual* for more information.

Endpoint Adapters

1. Use the **wep ls** command to make sure that the endpoint appears under the Tivoli Management Framework gateway you want. See the *IBM Tivoli Enterprise Console Reference Manual* for more information. Also make sure that any Tivoli Management Framework gateway the endpoint can log on to has ACF installed.
2. Source the endpoint environment and edit the **last.cfg** file in **$LCF_DATDIR**. Set log_threshold to 3 and then stop and restart the endpoint to enable endpoint tracing to the **lcfd.log** file. Check to make sure that the endpoint logged into an appropriate Tivoli Management Framework gateway.

3. If the endpoint has logged into a Tivoli Management Framework gateway successfully, create and distribute the ACP profile (see the *IBM Tivoli Enterprise Console User's Guide* for details). Check the **lcfd.log** file if there are further problems; you can also turn on tracing at the Tivoli Management Framework gateway and look in **$DBDIR/gatelog** for further debugging information.
4. If events do not arrive at the event server but are not incorrectly parsed, check to see if the events are caching on the endpoint instead. If so, either the **lcfd** process cannot communicate to the Tivoli Management Framework gateway or the event server, or the **lcfd** process itself is down. Verify that all communications among the event server, Tivoli Management Framework gateway, and endpoint are working.
5. Source the endpoint environment, then use the endpoint **wpostemsg** command from the system the adapter is running on to see if the event arrives at the event server. See the *IBM Tivoli Enterprise Console Reference Manual* for more information.

Non-TME Adapters

Use the **postemsg** command from the system on which the adapter is running to see if the event arrives at the event server. The **postemsg** command works in environments where Tivoli software is *not* installed. Thus, this standalone command displays error messages in English only, because the command does not have access to the message catalogs for the language support packs. See the *IBM Tivoli Enterprise Console Reference Manual* for more information.

Chapter 2. AS/400 Alert Adapter

The AS/400 alert adapter forwards events from an AS/400 system to the event server. The adapter can be registered with the startup configuration of the AS/400 so that the adapter is started with all the other applications when the system is started.

The AS/400 alert adapter is a program that does the following:

- Monitors AS/400 alert filters (using data queues) for alerts
- Extracts information from the alerts
- Creates IBM Tivoli Enterprise Console events, using a class definition statement (CDS) file
- Filters IBM Tivoli Enterprise Console events that are not important, using a configuration file
- Sends IBM Tivoli Enterprise Console events to an event server (using TCP/IP sockets) that runs user-created rules against these events

AS/400 alert events can be gathered from any alert filter, or from the supplied default filter. Multiple AS/400 alert adapters can be running at the same time, each monitoring a different filter.

A few of the benefits are as follows:

- Consolidates alert monitoring
- Integrates with existing AS/400 alert filters already defined to your specific business rules
- Filters out SNA (Systems Network Architecture) alerts that are not important and only notifies the Tivoli operators when something critical happens
- Automatically acts on events using customer defined rules and tasks (using the event server)
- Centrally configures adapter files that can be sent to the remote AS/400s

Adapter Files

The AS/400 alert adapter package consists of the following files:

/QSYS.LIB/QUSRSYS.LIB/CFG_ALERT.FILE/ALRCFG.MBR
The configuration file

/QSYS.LIB/QUSRSYS.LIB/CFG_ALERT.FILE/ALRCDS.MBR
The CDS file

/QSYS.LIB/QUSRSYS.LIB/CFG_ALERT.FILE/ALRBRC.MBR
The BAROC file

/QSYS.LIB/QUSRSYS.LIB/CFG_ALERT.FILE/ALRRLS.MBR
The rules file

Make a backup copy of the **CFG_ALERT** file before modifying the contents of any of the members.

A backup copy of this file also resides in the **CFG_ALERT** file in library **QTMETECA02**.

The AS/400 adapter package also consists of the following commands, which are copied into **QSYS** upon installation of the product:

STRTECADP Starts an AS/400 adapter.

ENDTECADP
Ends an AS/400 adapter.

Before starting the event server and an AS/400 alert adapter, check the configuration file to determine if it defines the preferred adapter behavior.

Configuration File

The configuration file for the AS/400 alert adapter defines the behavior of the adapter, which runs as a job on the AS/400.

A configuration file is created during the installation of the AS/400 alert adapter. The name of this file is **/QSYS.LIB/QUSRSYS.LIB/CFG_ALERT.FILE/ALRCFG.MBR**. The only keyword that is required to be set is **ServerLocation**. All other keywords have default values that are used if values are not specified.

The configuration file can contain the common keywords described in "Configuration File" on page 9, as well as the following adapter-specific keywords:

AdapterType Specifies the type of resource to be monitored. The default value is **MSGQ** if this keyword is not defined, meaning that the adapter monitors a message queue. The value provided in the configuration file is **ALERT**.

AdapterCdsFile
Specifies the CDS file to be used for the AS/400 alert adapter. This file can reside in either the **QSYS** or **IFS** name space, but the path must be specified in **IFS** notation, for example:

```
/QSYS.LIB/mylib.LIB/myfile.FILE/mymbr.MBR
```

The default is the following:

```
/QSYS.LIB/QUSRSYS.LIB/CFG_ALERT.FILE/ALRCDS.MBR
```

BufEvtPath Specifies the path and name of the buffer file for the AS/400 alert adapter. The default path is **/etc/Tivoli/tec**, and the default buffer file name is the value specified for the adapter name on the AS/400 command (**STRTECADP**), used to start the adapter.

Note: If an AS/400 alert adapter attempts to open a buffer file that is in use by another adapter, the adapter (which runs as a batch job) attempting to open the file ends.

Filter The name of the AS/400 alert filter to be monitored. The default value is **QTMETECA02/QYAAFTR**.

FilterDataQueue
The specific data queue that the adapter is to monitor for incoming alerts. If the alert filter is registered with the system, this keyword is required and the data queue must be created by the user before the AS/400 alert adapter is started. This keyword is optional if the alert filter defined by the **Filter** keyword is not registered with the system, or if the **Filter** keyword is not specified.

JobDescription
Specifies an AS/400 job description that is to be used when starting the adapter. The default is **QGPL/QDFTJOBD**.

LanguageID Specifies the AS/400 language ID in which alerts are to be sent to the event server. If a value is specified for this keyword, the AS/400 secondary language must be installed for that language ID. The default value for this keyword is **ENU**.

ProcessExistingAlerts
Specifies whether to send existing alerts on the data queue defined by the **FilterDataQueue** keyword. **NO** sends any new alerts sent to the data queue. **YES** sends the next alert received on the data queue. This can cause the adapter to resend previously sent alerts and create duplicate events sent to the event server. The default is **NO**.

ServerCCSID Specifies the coded character set identifier (CCSID) of the event server. This is in case the event server has a special code page or graphic character set that needs to be supported. The default is 00819.

Class Definition Statement File

The CDS file defines how events are constructed from information sent by the AS/400 alert adapter. It is described in detail in "Class Definition Statement File" on page 18.

SELECT Statement Example

```
SELECT
  1:ATTR(=,$ALERT_CDPT),VALUE(PREFIX, "10"); # 10xx codepoints
```

Here, **$ALERT_CDPT** is a custom keyword set by the adapter. These keywords can be used to write shorthand notation for SELECT statements. The following is equivalent to the previous example:

```
SELECT
  1:$ALERT_CDPT=10";
```

FETCH Statement Example

```
FETCH
  1:SUBSTR(value, start, length);
```

Keywords

To customize events, the AS/400 alert adapter supports the following keywords in class definition statements. Evaluation of these keywords is faster because access of them is direct. Event definition content and syntax are described in the *IBM Tivoli Enterprise Console Rule Builder's Guide.*

$ACTIONS Recommended actions to be taken for the alert.

$ACTION_CODE
The legacy action code for non-generic alerts (**alert subvector x'91'**).

$ADAPTER_CORREL
Unique alert identifier used to extract the alert from the alert database on the AS/400 system.

$ADAPTER_HOST
The protocol address of the host where the adapter is running.

$ADAPTER_HOST_SNANODE
The **netID.nau** name of the host where the adapter is running.

$ALERT_CDPT
The alert code point that provides an index into predefined text describing the alert condition.

$ALERT_ID The unique ID describing the alert.

$ARCH_TYPE Defines the alert type, either **NONGENERIC_ALERT** (**alert subvector x'91'**) or **GENERIC_ALERT** (**alert subvector x'92'**).

$BLOCK_ID The legacy block ID for non-generic alerts (**alert subvector x'91'**).

$CAUSES Alert causes collected from alert subvectors **x'93'**, **x'94**', **x'95'**, **x'96'**, and **x'97'**.

$DATE The date and time the event was generated.

$DETAILED_DATA
Product specific detail data from alert subvector **x'98'**.

$EVENT_CORREL
Alert correlation data from alert subvector **x'47'**.

$EVENT_TYPE
A value indicating the severity of the alert condition (for example, **PERMANENT**, **TEMPORARY**, or **IMPENDING PROBLEM**).

$HOSTNAME
The **netID.nau** name of the host where the alert originated.

$INCIDENT_CORREL
Alert correlation data from alert subvector **x'4A'**.

$MSG The alert code point text and the first probable cause text for the alert.

$ORIGIN The hierarchy list of the alert origin.

$PRODUCT_ID
The hardware and software identifier from alert subvector **x'10'**.

$SELF_DEF_MSG
The general message text from alert subvector **x'31'**.

$SEVERITY The severity of the event.

$SOURCE The source of the event. The source is defined by the adapter type **AS400_ALERT**.

$SUB_ORIGIN
The last member in the hierarchy list of the alert origin.

Configuring the AS/400 Alert Filters

Default Alert Filter

The AS/400 alert adapter creates a default alert filter, **QTMETECA02/QYAAFTR**, at installation time. This filter consists of a selection entry that maps all alerts to the group **QTECALERT**. The corresponding action entry for **QTECALERT** is also provided. When the AS/400 alert adapter is started, a data queue is created and the **QTECALERT** action entry is updated with the data queue name so incoming alert information can be monitored by the adapter.

If you use the default filter provided, copy it into library **QUSRSYS** and modify it there.

Integrating with an Existing Alert Filter

You might have alert filters that are already in use on your AS/400 system. These filters have been set up with the appropriate selection and action entries to filter alerts of interest and route them to predefined groups.

The **Filter** keyword in the configuration file is used to indicate the name of the filter that the AS/400 alert adapter is to monitor. If a value for this keyword is not specified, the default filter (**QTMETECA02/QYAAFTR**) is used.

The **FilterDataQueue** keyword in the configuration file is used to indicate the name of the data queue that the adapter is to monitor. The adapter assumes that this data queue has been created properly and has been incorporated into the appropriate action entries data queue list for the filter defined by the **Filter** keyword. To update an action entry, use the **CHGALRACNE** (**Change Alert Action Entry**) command. Create the data queue with the **Create Data Queue (CRTDTAQ**) command as follows:

```
CRTDTAQ DTAQ(library/name) TYPE(*STD) MAXLEN(592)
        FORCE(*NO) SEQ(*FIFO)
```

Note: If the data queue is not created per the previous specifications, the adapter will not start. Also, if the AS/400 alert adapter is not running, the system still sends alert information to this data queue. If the data queue is filled to capacity, the filter might be automatically deregistered by the system. To prevent this problem, have the adapter automatically started by a startup program **when the system is started** (see “Starting the Adapter” on page 27).

The AS/400 Network Attributes define the filter that is registered with the system. If the specified alert filter is registered with the system, then the **FilterDataQueue** keyword is required. If the filter is not registered with the system and the **FilterDataQueue** keyword is not specified, then a data queue is created and associated with the **QTECALERT** group in that filter. Use the **Change Network Attributes (CHGNETA)** command if you want to register the filter on the AS/400 system.

Starting the Adapter

The AS/400 adapter includes the **STRTECADP** command that enables you to start an adapter. You can also automatically start the adapter; see “Starting an AS/400 Adapter after an IPL” on page 35. The command is described on the following pages.

STRTECADP

Starts an AS/400 adapter.

SYNOPSIS

STRTECADP EVTADP(*name*) **CFGFILE**(*filename*)

DESCRIPTION

The AS/400 adapter runs as a batch job. The **STRTECADP** command starts an AS/400 adapter.

Authorization

QSYSOPR
***USE**

PUBLIC
***EXCLUDE**

Note: To grant other users authority to this command, use the following commands on the AS/400 system:

```
GRTOBJAUT OBJ(QSYS/STRTECADP) OBJTYPE(*CMD) USER(user) AUT(*USE)
GRTOBJAUT OBJ(QTMETECA/SBMEVTADAP) OBJTYPE(*PGM) USER(user) AUT(*USE)
GRTOBJAUT OBJ(QTMETECA02/STARTALERT) OBJTYPE(*PGM) USER(user) AUT(*USE)
GRTOBJAUT OBJ(QSYS/QNMRRGF) OBJECTYPE(*PGM) USER(user) AUT(*USE)
GRTOBJAUT OBJ(QSYS/QNMRGFN) OBJECTYPE(*PGM) USER(user) AUT(*USE)
GRTOBJAUT OBJ(QSYS/QNMDRGFN) OBJECTYPE(*PGM) USER(user) AUT(*USE)
```

Arguments

EVTADP(*name***)**
Specifies a name for the adapter being started. This name is used on the **ENDTECADP** AS/400 command. It can be any valid AS/400 job name; however, each adapter running on the AS/400 system must have a unique name.

CFGFILE(*filename***)**
Specifies the full path name of the configuration file, in IFS format, to be used.

EXAMPLES

The following command starts an AS/400 alert adapter using the default configuration file.

```
STRTECADP EVTADP(ALERTADP)
          CFGFILE('/QSYS.LIB/QUSRSYS.LIB/CFG_ALERT.FILE/ALRCFG.MBR')
```

The following command starts the AS/400 alert adapter with the **/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/MYCFG.MBR** configuration file.

```
STRTECADP EVTADP(MYADP)
          CFGFILE('/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/MYCFG.MBR')
```

Stopping the Adapter

The AS/400 adapter includes the **ENDTECADP** command that enables you to stop adapters individually or to stop all started adapters. The command is described on the following pages.

ENDTECADP

Stops the AS/400 adapter.

Context

ENDTECADP EVTADP(*name* | ***ALL**) [**OPTION**(***CNTRLD** | ***IMMED**)] [**DELAY**(*seconds*)]

Comments

The AS/400 adapter runs as a batch job. The **ENDTECADP** command stops an AS/400 adapter.

Authorization

QSYSOPR
***USE**

PUBLIC
***EXCLUDE**

Note: To grant other users authority to this command, use the following commands on the AS/400 system:

```
GRTOBJAUT OBJ(QSYS/ENDTECADP) OBJTYPE(*CMD) USER(user) AUT(*USE)
GRTOBJAUT OBJ(QTMETECA/ENDEVENTAD) OBJTYPE(*PGM) USER(user) AUT(*USE)
```

Arguments

EVTADP Specifies the name of the adapter to stop. The following options can be specified:

name Specifies the name of the adapter being stopped. This file name matches the name specified on the **STRTECADP** command.

***ALL** If ***ALL** is specified, then all adapters of all types are stopped.

OPTION Specifies the way the adapter stops. The following options can be specified:

***CNTRLD**
The adapter ends in a controlled manner. This lets the application program perform end-of-job processing.

***IMMED**
The adapter is ended immediately.

Stopping the adapter immediately does not allow the adapter to perform cleanup routines and is not recommended.

DELAY(*seconds***)**
Specifies the amount of time in seconds allowed for the adapter to complete its cleanup processing during a controlled end. This parameter is not used if ***IMMED** is specified for the **OPTION** parameter. If the cleanup is not completed before the end of the delay time, the adapter is ended immediately.

Examples

The following command stops the AS/400 alert adapter, started with the adapter name **ALERTADP.**

```
ENDTECADP EVTADP(ALERTADP)
```

The following command stops the AS/400 alert adapter, started with the adapter name **MYCFG**, in a controlled manner with a delay time of 60 seconds.

```
ENDTECADP EVTADP(MYCFG) OPTION(*CNTRLD) DELAY(60)
```

Events Listing

The following shows the class names and severities of all events defined for the AS/400 alert adapter. You can use it to get a sense of how AS/400 alert events are mapped to IBM Tivoli Enterprise Console events and to determine if you want to make any changes. The events are defined in the **tecad_snaevent.baroc** file on the event server.

See the *IBM Tivoli Enterprise Console Rule Builder's Guide* for more information about customizing the BAROC file.

Event Class Structure

Event classes are defined hierarchically, with child classes inheriting attribute value defaults from the parent. The AS/400 alert event classes follow a simple hierarchy. The attribute value for **source** is AS400_MSGQ. The following events are defined in the sample BAROC file provided with this product:

Event Class	**Default Event Severity**
AS400_TEC_ALERT_ADAPTER	(based on AS/400 alert type)
SNA_Event	CRITICAL
SNA_1xxx_Hardware	CRITICAL
SNA_Equipment_Malfunction	CRITICAL
SNA_Input_Device_Error	CRITICAL
SNA_Output_Device_Error	CRITICAL
SNA_Input_Output_Device_Error	CRITICAL
SNA_Loss_Of_Electrical_Power	CRITICAL
SNA_Loss_Of_Equipment_Cooling_Or_ Heating	CRITICAL
SNA_Subsystem_Failure	CRITICAL
SNA_Hardware	CRITICAL
SNA_2xxx_Software	CRITICAL
SNA_Software_Program_Abnormally_ Terminated	CRITICAL
SNA_Software_Program_Error	CRITICAL
SNA_Software_Operation_Failure	CRITICAL
SNA_Software	CRITICAL
SNA_3xxx_Communications	CRITICAL
SNA_Communication_Protocol_Error	CRITICAL
SNA_SNA_Protocol_Error	CRITICAL
SNA_LAN_Error	CRITICAL
SNA_Link_Error	CRITICAL
SNA_ISDN_Error	CRITICAL
SNA_Local_Connection_Error	CRITICAL
SNA_Link_Connection_Error	CRITICAL
SNA_BBNS_Communications_Error	CRITICAL
SNA_Communications	CRITICAL

Event Class			Default Event Severity
	SNA_4xxx_Performance		CRITICAL
		SNA_Performance_Degraded	CRITICAL
		SNA_Performance	CRITICAL
	SNA_5xxx_Congestion		CRITICAL
		SNA_Congestion	CRITICAL
		SNA_Configurable_Capacity_Limit_Reached	CRITICAL
		SNA_Congestion_Other	CRITICAL
	SNA_6xxx_Microcode		CRITICAL
		SNA_Microcode_Program_Abnormally_ Terminated	CRITICAL
		SNA_Microcode_Program_Error	CRITICAL
		SNA_Microcode_Program_Mismatch	CRITICAL
		SNA_Microcode	CRITICAL
	SNA_7xxx_Operator		CRITICAL
		SNA_Operator_Procedural_Error	CRITICAL
		SNA_Operator	CRITICAL
	SNA_8xxx_Specification		CRITICAL
		SNA_Configuration_Or_Customization_Error	CRITICAL
		SNA_Specification	CRITICAL
	SNA_9xxx_Intervention_Required		CRITICAL
		SNA_Operator_Intervention_Required	CRITICAL
		SNA_Stock_Low	CRITICAL
		SNA_Stock_Exhausted	CRITICAL
		SNA_Depository_Full	CRITICAL
		SNA_Intervention_Required	CRITICAL
	SNA_Axxx_Problem_Resolved		CRITICAL
		SNA_Problem_Resolved	CRITICAL
	SNA_Bxxx_Notification		CRITICAL
		SNA_Operator_Notification	CRITICAL
		SNA_Environmental_Problem	CRITICAL
		SNA_Resent_Alert_With_Updated_Information	CRITICAL
		SNA_Notification	CRITICAL
	SNA_Cxxx_Security		CRITICAL
		SNA_Security_Event	CRITICAL
		SNA_Security	CRITICAL
	SNA_Exxx_Non_IBM_Codepoint		CRITICAL
	SNA_Fxxx_Undetermined		CRITICAL
		SNA_Undetermined_Error	CRITICAL
	SNA_NonGeneric_Undetermined		CRITICAL
	SNA_Reserved_By_IBM		CRITICAL

You can set the severity of an AS/400 alert event on the event console as follows, based on the AS/400 alert type field specified in the message description:

Alert Type	Default Severity
01 (permanent loss of availability)	CRITICAL
04 (operator intervention required)	CRITICAL
09 (unavailable network component)	CRITICAL
0E (security problem)	CRITICAL
10 (permanently affected resource)	CRITICAL
03 (performance degradation)	WARNING
0A (notification: loss impending)	WARNING
0C (installation consistency)	WARNING
0D (operational procedural error)	WARNING
0F (delayed condition)	WARNING
11 (impending problem)	WARNING
14 (bypassed loss of availability)	WARNING
16 (monitored situation event)	WARNING
0B (environmental problem)	MINOR
12 (unknown)	UNKNOWN
02 (temporary loss of availability)	HARMLESS
05 (reserved)	HARMLESS
06 (reserved)	HARMLESS
07 (reserved)	HARMLESS
08 (reserved)	HARMLESS
13 (retired)	HARMLESS
other values	HARMLESS

Troubleshooting the AS/400 Adapter

If a problem occurs with the AS/400 adapter, you can perform problem determination by investigating the job the adapter is running in. Each time you start an AS/400 adapter, a batch job is started. You can view the adapter job by issuing the following command:

```
WRKJOB JOB(name)
```

Where *name* is the name of the adapter job that matches the name specified on the **STRTECADP** command. This will display the **Work with Job** dialog.

Note: Several adapter jobs might have existed on your AS/400 system with the same name as the current adapter job. In this case, you are first presented with a list of jobs to choose from. Select the most recent job from the list.

From the **Work with Job** dialog, you can select option 10 to display the job log, or if the job has ended (selecting option 10 will tell you so), you can view the job log that was generated by selecting option 4.

Examine the job log for messages indicating the error that occurred and follow the corrective action specified. For further assistance, contact Customer Support.

Logging Events in Test Mode

The file to which events are logged in test mode (instead of being sent to an event server) is created with a record length of 240 bytes if it does not exist. Because an event written to this file does not wrap to a new line if it is longer than 240 bytes, it is truncated. To avoid truncation, create the file ahead of time using the **CRTPF** or **CRTSRCPF** commands and specify a large enough record length to accommodate your events. To utilize this file, ensure that it is specified for the **ServerLocation** keyword. For additional information, see the **ServerLocation** and **TestMode** keywords on pages 13 and 14, respectively.

Also, be sure that you use the proper format, **ABCLIB/TECMSGS** (Library/Filename). If the file does not exist, it is created automatically.

TCP/IP Considerations

Ensure that the event server and the AS/400 are configured in your network Name Server, and that the AS/400 is configured to resolve to the Name Server.

If you do not use a Name Server in your network, make sure that an entry exists on the AS/400 in the TCP/IP host table for both the event server and the AS/400 system. Use the following commands to do this:

```
ADDTCPHTE INTNETADR('event server protocol address')
          HOSTNAME((event server host name))
          TEXT('Tivoli Enterprise Console event server')
ADDTCPHTE INTNETADR(AS/400 protocol address)
          HOSTNAME((AS/400 host name)) TEXT('AS/400')
```

Starting an AS/400 Adapter after an IPL

There are two methods that can be used to start an AS/400 alert adapter automatically after an initial program load (IPL), as follows:

- Adding an autostart job to a job queue
- Modifying the AS/400 startup program to call the **STRTECADP** command

Adding an Autostart Job to QSYSWRK

1. Create a Control Language (CL) program that will invoke the **STRTECADP** command, for example:
   a. Edit a source file member to add CL statements:
      ```
      STRSEU QGPL/QCLSRC STRADPCL
      ```
   b. Enter the following in the source file member. You can have a **STRTECADP** command for each adapter you would like to start:
      ```
      PGM
         STRTECADP  EVTADP(NEWFILTER) +
         CFGFILE('/QSYS.LIB/QUSRSYS.LIB/CFG_ALERT.FILE/ALRCFG.MBR')
      ENDPGM
      ```

      Note: Ensure that the TCP/IP service is started on the AS/400 system before starting an adapter.
   c. Create the program using the previous source program:
      ```
      CRTCLPGM  PGM(QGPL/STRADPCL) SRCFILE(QGPL/QCLSRC)
      ```
2. Create a job description that calls the previous program and use **QSYSNOMAX** as the job queue:

```
CRTJOBD JOBD(QGPL/STARTADP)
JOBQ(QSYSNOMAX)
TEXT('Start TEC adapter after IPL.')
RQSDTA('CALL QGPL/STRADPCL')
```

3. Add an auto start job entry in **QSYSWRK** using the previous job description:

```
ADDAJE SBSD(QSYSWRK)
JOB(TECAMSGQ)
JOBD(QGPL/STARTADP)
```

This program runs at the start of **QSYSWRK** subsystem and ends quickly after doing the **STRTECADP** command.

Changing the AS/400 Startup Program

The system value **QSTRUPPGM** (startup program) contains the name of the program to execute after IPL. This program can be modified to add the starting of adapters.

1. Retrieve the code in the startup program:

```
RTVCLSRC PGM(QSYS/program-name) SRCFILE(QGPL/QCLSRC)
         SRCMBR(program-name)
```

2. Modify the source:

```
PGM
   DCL VAR(&STRWTRS) TYPE(*CHAR) LEN(1)
   DCL VAR(&CTLSBSD) TYPE(*CHAR) LEN(20)
   QSYS/STRSBS SBSD(QCMN)
   STRTCP
   MONMSG MSGID(CPF0000)
   QSYS/STRSBS SBSD(QSERVER)
   MONMSG MSGID(CPF0000)

   STRTECADP EVTADP(ALERTADP)+
   CFGFILE('/QSYS.LIB/QUSRSYS.LIB/CFG_ALERT.FILE/ALRCFG.MBR')

   MONMSG MSGID(CPF0000)
   DONE:
   RETURN
   CHGVAR VAR(&CPYR) VALUE(&CPYR)
ENDPGM
```

3. Create the program and put it in the **QSYS** library:

```
CRTCLPGM PGM(QSYS/program-name) SRCFILE(QGPL/QCLSRC)
         SRCMBR(program-name)
```

Note: The startup program runs under user profile **QPGMR**. By default, **QPGMR** does not have authority to the AS/400 alert adapter commands and programs. You must either grant **QPGMR** authority to the commands and programs ("Starting the Adapter" on page 27) or have the startup program adopt **QSECOFR** authority and be owned by **QSECOFR**.

Multiple AS/400 Alert Adapters

To support another AS/400 alert adapter to monitor a different alert filter or another data queue within the same filter, create the following additional files:

- Configuration file: Specifies the filter to monitor and data queue to monitor.
- CDS file: Defines new classes to match the alerts being monitored.
- BAROC file: Required if new classes are identified in the CDS file.
- Rules file: Required if new rules are added.

Configuration File

To create the configuration file, perform the following steps:

1. Copy the adapter files using the following commands:

```
CPYF FROMFILE(QUSRSYS/CFG_ALERT)
     TOFILE(QUSRSYS/MYFILE) FROMMBR(*ALL)
     TOMBR(*FROMMBR) CRTFILE(*YES)
```

2. Update the configuration file to show the keywords pointing to the new objects, as follows:

```
AdapterCdsFile=/QSYS.LIB/QUSRSYS.LIB/MYFILE.FILE/MYCFG.MBR

Filter=mylib/myfilter

FilterDataQueue=mylib/mydtaqueue
```

3. Update the CDS and the BAROC files to include any new classes and filters.
4. Update the rules file to include any new rules.
5. On the event server, import the BAROC file into the rule base; then, compile and load the rule base.
6. Start the adapter using the new adapter files as follows:

```
STRTECADP EVTADP(MYEVTADP)
          CFGFILE('/QSYS.LIB/QUSRSYS.LIB/MYFILE.FILE/MYCFG.MBR')
```

POSTEMSG

Posts an event to the event server. See the *IBM Tivoli Enterprise Console Reference Manual* for more details about this command.

Context

QTMETECA/POSTEMSG { **–S**<*server*> | **–f**<*config_file*> } [**–r**<*severity*>] [**–m**<*message*>] [<*slot_name=value*>, ...] <*class*> <*source*>

Note: There cannot be a space between the option letter and the option value.

Examples

```
Call QTMETECA/POSTEMSG PARM('–Sserver_name' '–rHARMLESS'
     '–m"This is a message"' AS400_MSG LOGFILE)
Call QTMETECA/POSTEMSG
     PARM('–f/QSYS.LIB/QUSRSYS.LIB/CFG_MSG.FILE/MSGCFG.MBR'
     '–rFATAL' '–m"This is a message"' AS400_MSG LOGFILE)
```

Chapter 3. AS/400 Message Adapter

The AS/400 message adapter forwards events from an AS/400 system to the event server. It can be registered with the startup configuration of the AS/400 system so that the adapter is started with all the other applications when the AS/400 system is started. See "Starting an AS/400 Adapter after an IPL" on page 52 for instructions on starting the adapter automatically with the AS/400 system.

The AS/400 message adapter is a program that does the following:

- Reads messages from a message queue on an AS/400 system
- Extracts information from the message
- Creates IBM Tivoli Enterprise Console classes, using a class definition statement (CDS) file
- Filters IBM Tivoli Enterprise Console events that are not important, using a configuration file
- Sends IBM Tivoli Enterprise Console events to an event server (using TCP/IP sockets) that runs user-created rules against these events

AS/400 message events can be gathered from any non-program message queue, including the system operator message queue QSYSOPR. Multiple AS/400 message adapters can be running at the same time. One AS/400 message adapter can monitor the system operator message queue while another is monitoring an application message queue.

A few of the benefits of the AS/400 message adapter are as follows:

- Consolidates the system operator message console, QSYSOPR, for all the AS/400 systems in your enterprise
- Monitors applications that use message queues
- Filters out messages that are not important and only notifies the Tivoli operators when something critical happens
- Automatically acts on events using customer-defined rules and tasks (using the event server)
- Centrally configures adapter files that can be sent to remote AS/400 systems

Adapter Files

The AS/400 adapter package consists of the following files:

/QSYS.LIB/QUSRSYS.LIB/CFG_MSG.FILE/MSGCFG.MBR
The configuration file.

/QSYS.LIB/QUSRSYS.LIB/CFG_MSG.FILE/MSGCDS.MBR
The CDS file.

/QSYS.LIB/QUSRSYS.LIB/CFG_MSG.FILE/MSGBRC.MBR
The BAROC file. This file is located on the event server with the name of **as400msg.baroc**. It is automatically compiled into the active rule base when the event server is installed.

Make a backup copy of the **CFG_MSG** file if you intend to modify the contents of any of the members.

A backup copy of each of these files also resides in the **CFG_MSG** file in library **QTMETECA01**.

Before starting the event server and an AS/400 message adapter, check the configuration file to determine if it defines the preferred adapter behavior.

Configuration File

The configuration file for the AS/400 message adapter defines the behavior of the adapter, which runs as a job on the AS/400 system.

A configuration file is created during the installation of the AS/400 message adapter. The name of this file is **/QSYS.LIB/QUSRSYS.LIB/CFG_MSG.FILE/MSGCFG.MBR**. The configuration file can contain the keywords described in "Configuration File" on page 9, as well as the following custom keywords:

AdapterType Specifies the type of resource to be monitored. The default value is **MSGQ**, meaning that the adapter monitors a message queue.

AdapterCdsFile
Specifies the CDS file to be used for the AS/400 message adapter. This file can reside in either the **QSYS** or **IFS** name space, but the path must be specified in **IFS** notation, for example:

```
/QSYS.LIB/mylib.LIB/myfile.FILE/mymbr.MBR
```

The default is the following:

```
/QSYS.LIB/QUSRSYS.LIB/CFG_MSG.FILE/MSGCDS.MBR
```

BufEvtPath Specifies the path and name of the buffer file for the AS/400 message adapter. The default path is **/etc/Tivoli/tec**, and the default buffer file name is the value specified for the adapter name on the AS/400 command (**STRTECADP**), used to start the adapter.

Note: If an AS/400 message adapter attempts to open a buffer file that is in use by another adapter, the adapter (which runs as a batch job) attempting to open the file ends.

JobDescription
Specifies an AS/400 job description that is to be used when starting the adapter. The default is **QGPL/QDFTJOBD**.

LanguageID Specifies the AS/400 language ID in which the AS/400 messages are to be sent to the event server. The default value for this keyword is **ENU**. If a value is specified for this keyword, the AS/400 secondary language must be installed for that language ID.

MsgQueue Specifies the AS/400 message queue to poll. The complete name needs to be specified. The message queue must exist when the adapter is started. If the message queue is cleared while the adapter is active, the adapter starts with new messages that are written after the message queue was cleared. The value of this field must be in the following format:

```
mylib/mymsgq
```

The default is **QSYS/QSYSOPR**.

PollInterval Specifies the amount of time in seconds to return to a suspended state between checking for new events that have been placed on the message queue. The default is 20. The following example shows the format:

```
PollInterval=60
```

ProcessExistingMsgs
Specifies whether the AS/400 messages adapter resets back to the first message on the message queue when starting. **NO** sends any new messages to the message queue. **YES** sends the first message on the message queue. This could cause the adapter to resend previously sent messages and create duplicate events sent to the event server. The default is **NO**.

ServerCCSID Specifies the coded character set identifier (CCSID) of the event server. This is in case the event server has a special code page or graphic character set that needs to be supported. The default is 0819.

Class Definition Statement File

The file **/QSYS.LIB/QUSRSYS.LIB/CFG_MSG.FILE/ MSGCDS.MBR** defines how events are constructed from information sent by the AS/400 message adapter. It is described in detail in "Class Definition Statement File" on page 18.

SELECT Statement Example

```
SELECT
  1:ATTR(=,$MSG_ID), VALUE(=,CPI5933);
```

Here, **$MSG_ID** is a custom keyword set by the adapter. These keywords can be used to write shorthand notation for SELECT statements. The following is equivalent to the previous example:

```
SELECT
  1:$MSG_ID=CPI5933;
```

For the **$MSG_ID** keyword, multiple low:high pairs can be specified with spaces as separators. An example is as follows:

```
SELECT
  1:$MSG_ID=CPF 0100:02FF 1000:1FFF 5600:56FF;
```

FETCH Statement Example

```
FETCH
  1:SUBSTR(value, start, length);
```

MAP Statement Example

```
CLASS PerformanceInvestigator
SELECT
  1:$MSG_ID=PNV *:*;
FETCH
  1:SUBSTR($V1, 0, 3);
  2:SUBSTR($V1, 3, 4);
MAP
  my_field=PRINTF("attribute=%s has prefix=%s and id=%s", $V1,
$F1, $F2);
  status=OPEN
END
```

Keywords

To customize events, the AS/400 message adapter supports the following keywords in class definition statements. Evaluation of these keywords is faster

because access of them is direct. Event definition content and syntax are described in the *IBM Tivoli Enterprise Console Rule Builder's Guide*.

$ADAPTER_HOST
The protocol address of the host where the adapter is running.

$ALERT_OPTION
If and when an SNA alert is created and sent for the message. If a message is received, the value is one of the following:

***DEFER**
An alert is sent after local problem analysis.

***IMMED**
An alert is sent immediately when the message is sent to the **QHST** message queue.

***NO** No alert is sent.

***UNATTEND**
An alert is sent immediately when the system is running in unattended mode (when the value of the alert status network attribute, **ALRSTS**, is ***UNATTEND**).

$DATE The date and time the event was generated.

$DATA_CCSID_CONVERT_STATUS
The following are possible values returned:

0 No conversion was needed because the CCSID of the replacement data or impromptu message text matched the CCSID you wanted the data or text converted to.

1 No conversion occurred because either the data was 65535 or the CCSID you wanted the data converted to was 65535.

2 No conversion occurred because you did not supply enough space for the data.

3 The data was converted to the CCSID specified using the best fit conversion tables.

4 A conversion error occurred using the best fit conversion tables, so a default conversion was attempted. This completed without error.

–1 An error occurred on both the best fit and default conversions. The data was not converted.

$DATA_CCSID_RETURNED
The CCSID of the replacement data or impromptu message text is returned. If an impromptu message is received, this is the CCSID of the impromptu message text. When replacement data is received, this is the CCSID of the replacement data fields defined as convertible character (***CCHAR**) in the message description. All other replacement data is not converted before it is returned. If a conversion error occurs or the CCSID you requested the data to be converted to is 65535, the CCSID of the data or text is returned. If replacement data is being returned and there is no ***CCHAR** replacement data, 65535 is returned. Otherwise, the CCSID you wanted the data converted to is returned.

$HOSTNAME
The name of the system on which the event occurred.

$MSG The default message used.

$MSG_FILE_NAME
The name of the message file containing the message received.

$MSG_FILE_LIBRARY
The name of the library containing the message file. For the actual library used when the message is sent, use the **$MSG_LIBRARY_USED** keyword.

$MSG_HELP The message help for the message received. If an immediate message is received, this field is blank.

$MSG_ID Indicates the AS/400 message identifier.

$MSG_KEY The key to the message received.

$MSG_LIBRARY_USED
The name of the library used to send the message. Because the library can contain override instructions, this is not necessarily the library in which the message actually resides.

$MSG_SEVERITY
Specifies the severity. A two-digit value ranging from 0 through 99. The higher the value, the more severe or important the condition.

$MSG_TYPE The message type of the message received. The possible values and their meanings are the following:

- **01** Completion
- **02** Diagnostic
- **04** Informational
- **05** Inquiry
- **06** Sender's copy
- **08** Request
- **10** Request with prompting
- **14** Notify
- **15** Escape
- **21** Reply, not validity checked
- **22** Reply, validity checked
- **23** Reply, message default used
- **24** Reply, system default used
- **25** Reply, from system reply list

$ORIGIN The protocol address of the source system.

$SEND_DATE
The date on which the message was sent, in CYYMMDD (century, year, month, day) format.

$SEND_JOB The name of the job in which the message being received was sent.

$SEND_JOB_NUMBER
The job number of the job in which the message being received was sent.

$SEND_PROGRAM_NAME
The program name or Integrated Language Environment® (ILE) program name that contains the procedure sending the message.

$SEND_TIME
The time at which the message being received was sent, in HHMMSS (hour, minute, second) format.

$SEND_USER_PROFILE
The name of the user profile that sent the message being received.

$SEVERITY The severity of the event.

$SOURCE The source of the event. The source is defined by the adapter type (**AS400_MSGQ**).

$SUB_ORIGIN
A further categorization of the origin.

$SUB_SOURCE
A further categorization of the source.

$TEXT_CCSID_CONVERT_STATUS
The following are possible values returned:

0 No conversion was needed because the CCSID of the message or message help text matched the CCSID you wanted the message or message help text converted to.

1 No conversion occurred because either the message or message help text was 65535 or the CCSID you wanted the message or message help text converted to was 65535.

2 No conversion occurred because you did not supply enough space for the message or message help.

3 The message or message help text was converted to the CCSID specified using the best fit conversion tables.

4 A conversion error occurred using the best fit conversion tables, so a default conversion was attempted. This completed without error.

–1 An error occurred on both the best fit and default conversions. The data was not converted.

$TEXT_CCSID_RETURNED
The CCSID of the text in the message and message help fields is returned. The inserted replacement data might not be the same CCSID. Refer to the **$DATA_CCSID_RETURNED** keyword for more details. If a conversion error occurs or the CCSID you requested the text to be converted to is 65535, the CCSID that the message description is stored in is returned. Otherwise, the CCSID you wanted your text converted to is returned. If you do not want the text converted before it is returned to you but you do want to know the CCSID that the message description is stored in, specify 65535 on the coded character set identifier parameter. The CCSID that the message description is stored in is returned in the CCSID of message and message help output field.

$ARG1 – $ARG8
Used to identify message replacement text or values.

Starting the Adapter

The AS/400 message adapter includes the **STRTECADP** command that enables you to start an adapter. The command is described on the following pages.

STRTECADP

Starts an AS/400 adapter.

Flags

STRTECADP EVTADP(*name*) **CFGFILE**(*filename*)

Comments

The AS/400 adapters run as a batch job. The **STRTECADP** command starts an AS/400 adapter.

Authorization

QSYSOPR
***USE**

PUBLIC
***EXCLUDE**

To grant other users authority to this command, use the following commands on the AS/400:

```
GRTOBJAUT OBJ(QSYS/STRTECADP) OBJTYPE(*CMD) USER(user) AUT(*USE)
GRTOBJAUT OBJ(QTMETECA/SBMEVTADAP) OBJTYPE(*PGM) USER(user) AUT(*USE)
GRTOBJAUT OBJ(QTMETECA01/STARTMSGAD) OBJTYPE(*PGM) USER(user) AUT(*USE)
```

Arguments:

EVTADP(*name***)**
Specifies a name for the adapter being started. This name is used on the **End TEC Adapter** (**ENDTECADP**) AS/400 command. It can be any valid AS/400 job name; however, each adapter running on the AS/400 system must have a unique name.

CFGFILE(*filename***)**
Specifies the full path name of the configuration file, in IFS format, to be used.

Examples

The following command starts an AS/400 message adapter using the default configuration file. The default configuration file monitors the system operator message queue, **QSYSOPR**:

```
STRTECADP EVTADP(SYSOPR)
          CFGFILE('/QSYS.LIB/QUSRSYS.LIB/CFG_MSG.FILE/MSGCFG.MBR')
```

The following command starts the AS/400 message adapter with the **/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/MYCFG.MBR** configuration file. The configuration file could be set up to monitor an application specific message queue:

```
STRTECADP EVTADP(MYAPP)
          CFGFILE('/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/MYCFG.MBR')
```

Stopping the Adapter

The AS/400 adapter includes the **ENDTECADP** command that enables you to stop adapters individually or to stop all started adapters. The command is described on the following pages.

ENDTECADP

Stops the AS/400 adapter.

Context

ENDTECADP EVTADP(*name* | ***ALL**) [**OPTION**(***CNTRLD** | ***IMMED**)] [**DELAY**(*seconds*)]

Comments

The AS/400 adapters run as a batch job. The **ENDTECADP** command stops an AS/400 adapter.

Authorization

QSYSOPR
***USE**

PUBLIC
***EXCLUDE**

To grant other users authority to this command, use the following commands on the AS/400:

```
GRTOBJAUT OBJ(QSYS/ENDTECADP) OBJTYPE(*CMD) USER(user) AUT(*USE)
GRTOBJAUT OBJ(QTMETECA/ENDEVENTAD) OBJTYPE(*PGM) USER(user) AUT(*USE)
```

Arguments

EVTADP
Specifies the name of the adapter to stop. The following options can be specified:

name Specifies the name of the adapter being stopped. This name matches the name specified on the **Start TEC Event Adapter** command.

***ALL** If ***ALL** is specified, then all adapters of all types are stopped.

OPTION
Specifies the way the adapter stops. The following options can be specified:

***CNTRLD**
The adapter ends in a controlled manner. This lets the application program perform end-of-job processing.

***IMMED**
The adapter is ended immediately.

Note: Stopping the adapter immediately does not allow the adapter to perform cleanup routines and is not recommended.

DELAY(*seconds***)**
Specifies the amount of time in seconds allowed for the adapter to complete its cleanup processing during a controlled end. This parameter is not used if ***IMMED** is specified for the **OPTION** parameter. If the cleanup is not completed before the end of the delay time, the adapter is ended immediately.

Examples

The following command stops the AS/400 message adapter, started with the adapter name **SYSOPR**, which was started to monitor the **QSYSOPR** message queue:

```
ENDTECADP EVTADP(SYSOPR)
```

The following command stops the AS/400 message adapter, started with the adapter name **MYAPP**, in a controlled manner that was set up to monitor an application-specific message queue:

```
ENDTECADP EVTADP(MYAPP) OPTION(*CNTRLD) DELAY(60)
```

Events Listing

The following shows the class names and severities of all events defined for the AS/400 message adapter. You can use it to get a sense of how AS/400 messages are mapped to IBM Tivoli Enterprise Console events and to determine if you want to make any changes. The events are defined in the **as400msg.baroc** file on the event server.

See the *IBM Tivoli Enterprise Console Rule Builder's Guide* for more information about customizing the BAROC file.

Event Class Structure

Event classes are defined hierarchically, with child classes inheriting attribute value defaults from the parent. The AS/400 message event classes follow a simple hierarchy. The AS/400 message adapter fills in the following attribute defaults. The attributes are used in event group filters.

source AS400_MSGQ

sub_source
Fully qualified message queue name.

origin Protocol address of the system.

hostname
Name of the system from the host name table.

date Date and time the message was sent.

msg First level message text with replacement values.

The following events are defined in the sample BAROC file provided with this product:

Event Class		Default Severity
AS400_TEC_MSGQ_ADAPTER		(Based on the AS/400 message severity) 00-19 HARMLESS 20-29 WARNING 30-39 MINOR 40-59 CRITICAL 60-99 FATAL
AS400_MSG_BASE		(Based on the AS/400 message severity) 00-19 HARMLESS 20-29 WARNING 30-39 MINOR 40-59 CRITICAL 60-99 FATAL
	AS400_MSG	
	AS400_Writer_Started	
	AS400_Writer_Ended_Normal	
	AS400_Device_No_Longer_Communicating	
	AS400_Controller_Failed	
	AS400_Controller_NotReplying	
	AS400_Network_Session_Unavailable	
	AS400_Controller_Contacted_Line	
	AS400_Controller_Off_or_NotRecognized	
	AS400_Unable_Auto_VaryOn	

Troubleshooting the AS/400 Adapter

If a problem occurs with the AS/400 adapter, you can perform problem determination by investigating the job the adapter is running in. Each time you start an AS/400 adapter, a batch job is started. You can view the adapter job by issuing the following command:

```
WRKJOB JOB(name)
```

Where *name* is the name of the adapter job that matches the name specified on the **STRTECADP** command. This displays the **Work with Job** dialog.

Note: Several adapter jobs might have existed on your AS/400 with the same name as the current adapter job. In this case, you are first presented with a list of jobs to choose from. Select the most recent job from the list.

From the **Work with Job** dialog, you can select option 10 to display the job log, or if the job has ended (selecting option 10 tells you so), you can view the job log that was generated by selecting option 4.

Examine the job log for messages indicating the error that occurred and follow the corrective action specified. For further assistance, contact Customer Support.

Logging Events in Test Mode

The file to which events are logged in test mode (instead of being sent to an event server) is created with a record length of 240 bytes if it does not exist. Because an event written to this file does not wrap to a new line if it is longer than 240 bytes, it is truncated. To avoid truncation, create the file ahead of time using the **CRTPF** or **CRTSRCPF** commands and specify a large enough record length to accommodate your events. To utilize this file, ensure it is specified for the **ServerLocation** keyword. For additional information, see the **ServerLocation** and **TestMode** keywords on pages 13 and 14, respectively.

Also, be sure that you use the proper format, **ABCLIB/TECMSGS** (Library/Filename). If the file does not exist, it is created automatically.

TCP/IP Considerations

Ensure that the event server and the AS/400 system are configured in your network Name Server, and that the AS/400 system is configured to resolve to the Name Server.

If you do not use a Name Server in your network, make sure that an entry exists on the AS/400 system in the TCP/IP host table for both the event server and the AS/400 system. Use the following commands to do this:

```
ADDTCPHTE INTNETADR(event server protocol address)
          HOSTNAME((event server host name))
          TEXT('Tivoli Enterprise Console event server')
ADDTCPHTE INTNETADR(AS/400 protocol address)
          HOSTNAME((AS/400 host name)) TEXT('AS/400')
```

Starting an AS/400 Adapter after an IPL

Two methods can be used to automatically start an AS/400 message adapter after an IPL:

- Adding an autostart job to a job queue
- Modifying the AS/400 start-up program to call the **STRTECADP** command

Adding an Autostart Job to QSYSWRK

1. Create a CL program that invokes the **STRTECADP** command, for example:
   a. Edit a source file member to add CL statements:
      ```
      STRSEU QGPL/QCLSRC STRADPCL
      ```
   b. Enter the following in the source file member. You can have a **STRTECADP** command for each adapter you would like to start:
      ```
      PGM
         STRTECADP EVTADP(SYSOPR) +
         CFGFILE('/QSYS.LIB/QUSRSYS.LIB/CFG_MSG.FILE/MSGCFG.MBR')
      ENDPGM
      ```

      Note: Ensure that TCP/IP service is started on the AS/400 system before starting a message adapter.
   c. Create the program using the previous source member:
      ```
      CRTCLPGM  PGM(QGPL/STRADPCL) SRCFILE(QGPL/QCLSRC)
      ```
2. Create a job description that calls the previous program and use **QSYSNOMAX** as the Job Queue:
   ```
   CRTJOBD JOBD(QGPL/STARTADP)
           JOBQ(QSYSNOMAX)
           TEXT('Start TEC adapter after IPL.')
           RQSDTA('CALL QGPL/STRADPCL')
   ```
3. Add an auto-start job entry in **QSYSWRK** using the previous job description:
   ```
   ADDAJE SBSD(QSYSWRK) JOB(TECAMSGQ) JOBD(QGPL/STARTADP)
   ```

   This program runs at the start of **QSYSWRK** subsystem and ends quickly after doing the **STRTECADP** command.

Changing the AS/400 Startup Program

The system value **QSTRUPPGM** (start-up program) contains the name of the program to execute after IPL. This program can be modified to add the starting of adapters.

1. Retrieve the code in the start-up program:
   ```
   RTVCLSRC PGM(QSYS/program-name) SRCFILE(QGPL/QCLSRC)
            SRCMBR(program-name)
   ```
2. Modify the source:
   ```
   PGM
      DCL VAR(&STRWTRS) TYPE(*CHAR) LEN(1)
      DCL VAR(&CTLSBSD) TYPE(*CHAR) LEN(20)
      QSYS/STRSBS SBSD(QCMN)
      STRTCP
      MONMSG MSGID(CPF0000)
      QSYS/STRSBS SBSD(QSERVER)
      MONMSG MSGID(CPF0000)
      STRTECADP  EVTADP(SYSOPR)+
      CFGFILE('/QSYS.LIB/QUSRSYS.LIB/CFG_MSG.FILE/MSGCDS.MBR')
      MONMSG MSGID(CPF0000)
   ```

```
   DONE:
   RETURN
   CHGVAR VAR(&CPYR) VALUE(&CPYR)
ENDPGM
```

3. Create the program and put it in the **QSYS** library:

```
CRTCLPGM PGM(QSYS/program-name)
         SRCFILE(QGPL/QCLSRC) SRCMBR(program-name)
```

Note: The start-up program runs under user profile **QPGMR**. By default, **QPGMR** does not have authority to change the AS/400 message adapter commands and programs. You must either grant **QPGMR** authority to change the commands and programs (see "Starting the Adapter" on page 45) or have the start-up program adopt **QSECOFR** authority and be owned by **QSECOFR**.

Multiple AS/400 Message Queues

To support another AS/400 message queue, create the following additional files:

- Configuration file: specifies a different message queue for the **MsgQueue keyword** and any new filters
- CDS file: defines new classes to match the messages being monitored
- BAROC file: required if new classes are identified in the CDS file

Configuration File

To create the configuration file, perform the following steps:

1. Copy the adapter files using the following commands:

```
CPYF FROMFILE(QUSRSYS/CFG_MSG)
     TOFILE(QUSRSYS/MYFILE) FROMMBR(*ALL)
     TOMBR(*FROMMBR) CRTFILE(*YES)
```

2. Update the configuration file to show the keywords pointing to the new objects as follows:

```
AdapterCdsFile=/QSYS.LIB/QUSRSYS.LIB/MYFILE.FILE/MYCDS.MBR

MsgQueue=QUSRSYS/MYMSGQ
```

3. Update the CDS and the BAROC files to include any new classes and filters.
4. On the event server, import the BAROC file into the rule base; then, compile and load the rule base.
5. Start the adapter using the new configuration files as follows:

```
STRTECADP EVTADP(MYEVTADP)
          CFGFILE('/QSYS.LIB/QUSRSYS.LIB/MYFILE.FILE/MYCFG.MBR')
```

Using FTP to Execute AS/400 Commands

You can execute AS/400 commands from an FTP session. This can be useful for replying to inquiry messages. The following is an example of how to use FTP to remotely respond to an AS/400 inquiry message based on the message key that is part of the event string:

```
quote "RCMD SNDRPY MSGKEY(X'00022A00') MSGQ(QSYSOPR) RPY('The reply') RMV(*NO)"
```

Chapter 4. NetWare Log File Adapter

The following sections contain reference information about the NetWare log file adapter.

NetWare Log File Adapter Reference Information

The log file adapter for NetWare forwards events from a NetWare server to the event server. The NetWare log file adapter can be registered with the startup configuration of the NetWare server so that the log file adapter is started when the NetWare server is started.

NetWare server events are gathered from any ASCII log file residing on the NetWare server, such as the **SYS:SYSTEM\SYS$LOG.ERR** file.

The NetWare log file adapter is a NetWare Loadable Module (NLM) process that reads events generated on a NetWare server, formats them according to specifications in the format file, and forwards them to the event server for further processing.

The NetWare log file adapter can run silently, without its own screen, or it can run in the debugging mode that displays screen messages for diagnostic purposes.

Adapter Files

The NetWare server adapter package consists of the following files:

tecadnw4.nlm
The adapter service executable file

tecadnw4.cnf
The configuration file

tecadnw4.cds
The class definition statement (CDS) file

tecadnw4.brc
The BAROC file

postmsg.nlm
The command line interface program to send an event to the event server

nwgencds.nlm
The command line interface program to generate a CDS file from a format file

tecadnw4.err
The error file

Before starting the server, ensure that the configuration file defines the preferred adapter behavior.

Error File

The error file enables you to configure debugging and tracing options. This file is described in detail in “Error File” on page 19.

Prefiltering NetWare Events

You can improve the performance of the NetWare log file adapter by filtering events, so that only important events are processed. This is called prefiltering and applies only to events logged to the **SYS$LOG.ERR** file.

To use the prefiltering mechanism, you specify the prefilter statements in the configuration file using a format similar to that used for adapter filters. The prefiltering statements (**PreFilter** and **PreFilterMode**) are described in "Configuration File" on page 56.

You must stop and restart the adapter for any changes to take effect.

The following attributes define prefilter statements:

Source
Specifies the source or module that logged the event to the NetWare server log file. You can specify up to 16 sources. Multiple sources must be separated by commas. Examples include SERVER, DS, TIMESYNC, and UPS.

EventId
Specifies the message number assigned by NetWare. You can specify up to 16 message numbers. Message numbers must be separated by commas. **EventId** is unique for each source.

Severity
Specifies the NetWare-defined severity of the event. You can specify up to 16 severities. Multiple severities must be separated by commas.

Locus Specifies the NetWare-defined locus. You can specify up to 16 loci. Multiple loci must be separated by commas.

Class Specifies the NetWare-defined class. You can specify up to 16 classes. Multiple classes must be separated by commas.

The following are examples of prefiltering statements:

```
PreFilter:Source=SERVER;EventId=10,20,30;
PreFilter:Source=DS; Severity=11;Class=5;
```

Configuration File

The configuration file defines the behavior of the NetWare log file adapter. This file can contain the common keywords listed in "Configuration File" on page 9, as well as the following adapter-specific keywords:

LogSources
Specifies the ASCII log files to poll for messages. The complete path to each file must be specified, and file names must be separated by commas; no spaces or other separators can be used. A log file source need not exist when the adapter is started; it is polled when it is created.

If a file is truncated while the adapter is active, the adapter automatically sets its internal pointer to the new end of the file and continues processing all new messages that are written after the file was truncated. If during the polling interval the file is overwritten, removed, or recreated with more lines than the previous poll, only the number of lines greater than the

previous line count is read. For example, the file has one line. After the poll interval elapses, the file is overwritten with two lines. Only the second line is read on the next polling.

The adapter polls the **SYS:SYTEM\SYS$LOG.ERR** file by default. Additional files can be specified with the **LogSources** keyword.

PollInterval

Specifies the frequency, in seconds, to poll each log file listed in the **LogSources** keyword for new messages. The default value is 120 seconds.

PreFilter

An event matches a **PreFilter** statement when each **attribute=value** specification in the **PreFilter** statement matches a message in the log file. A **PreFilter** statement must contain at least the log file specification, and can contain up to three additional specifications: event ID, event type, and event source. The order of the attributes in the statement does not matter.

You can specify multiple values for each attribute by separating each with a comma.

Each **PreFilter** statement must be on and contained in a single line, no greater than 512 characters.

The **PreFilter** keyword is optional. All NetWare server log events are sent to the adapter if prefilters are not specified.

PreFilterMode

Specifies whether NetWare server log events that match a **PreFilter** statement are sent (**PreFilterMode=IN**) or ignored (**PreFilterMode=OUT**). Valid values are **IN**, **in**, **OUT**, or **out**. The default is **OUT**.

The **PreFilterMode** keyword is optional; if **PreFilterMode** is not specified, only events that do not match any **PreFilter** statements are sent to the adapter.

If you set **PreFilterMode=IN**, make sure you have one or more **PreFilter** statements defined as well.

Stop and restart the adapter for any changes to take effect.

Format File

The format file contains message format descriptions and their mapping to BAROC events. The message fields of a NetWare server event are matched against the format descriptions in this file and when a match succeeds, the corresponding IBM Tivoli Enterprise Console event is generated by the adapter. The format file contains predefined mappings for some common NetWare server events and can be customized to add new messages.

A standard NetWare server event from the **SYS$LOG.ERR** file is written to an ASCII message in the following sequence. Consult the appropriate NetWare manuals for the meanings:

- The date (month-day-year) and time; for example: 7-25-98 1:33:57 am
- Module version-ID; for example: SERVER-4.11-25
- Severity, locus, and class; for example: Severity=10 Locus=1 Class=5

  Note: The meanings of *severity* and *class* are not the same as those pertaining to the IBM Tivoli Enterprise Console product.
- The message text

The following example shows a formatted IBM Tivoli Enterprise Console event derived from an error message issued by the NetWare Directory Service (DS):

```
7-16-98 5:08:46 pm:DS-5.73-12 Severity=10 Locus=2 Class=5
Synthetic Time is being issued on partition "NOVELL_TREE."
```

For details about format files, see “Format File” on page 17.

Events Listing

The tables in the next section show the class names and severities of all events defined for the NetWare log file adapter. You can use this information to get a sense of how NetWare events are mapped to IBM Tivoli Enterprise Console events and to determine whether you want to make any changes. The events are defined in the BAROC file, which must be imported into the rule base. See the *IBM Tivoli Enterprise Console Rule Builder's Guide* for more information about customizing the BAROC file.

Event Class Structure

Event classes are defined hierarchically, with child classes inheriting attribute value defaults from the parent. The NetWare server event classes follow a simple hierarchy. The adapter fills in the following attribute default values, as shown in the following table. The attributes are used in event group filters.

Attribute	Default Value
source	NW4
sub_source	NW4

When an event from the **SYS$LOG.ERR** file is sent, the **sub_source** attribute is set to the module that logged the event (for example, DS or SERVER). The default event classes define the following attributes:

nw_msg_version
This is the version of the module (**sub_source**) that is logging the message, for example, 4.10, 1.0, and so on.

nw_msg_id
This is an integer value specifying the message ID. A message ID is unique within each **sub_source**.

alert_severity
Specified as an integer from zero (0) to 6, this value indicates the severity level defined by NetWare. The mapping between the NetWare **alert_severity** and IBM Tivoli Enterprise Console severity level is defined in the following table.

Alert Severity	Definition	Severity Level
0 (Informational)	Counters or gauges reached thresholds.	HARMLESS
1 (Warning)	Configuration errors, and so on. No damage.	WARNING
2 (Recoverable)	Hot Fix, and so on. Workaround made.	MINOR
3 (Critical)	Disk Mirror failure, and so on. Fix attempted.	CRITICAL

Alert Severity	Definition	Severity Level
4 (Fatal)	Resource fatally affected; shutdown.	FATAL
5 (Operation Aborted)	The operation cannot complete.	FATAL
6 (Non OS unrecoverable)	The operation cannot complete.	FATAL

alert_locus

Specified as an integer from zero (0) to 20, this value indicates the location of the alert, as defined in the following table:

Alert_locus	NetWare Definition
0	Unknown
1	Memory
2	File system
3	Disks
4	Lanboards
5	Comstacks
7	TTS
8	Bindery
9	Station
10	Router
11	Locks
12	Kernel
13	UPS
14	Service Protocol
15	SFTIII
16	Resource Tracking
17	NLM
18	OS Information
19	Cache
20	Domain

alert_class

Specified as an integer from zero (0) to 21, this value indicates the NetWare alert classes as defined in the following table:

Alert_class	NetWare Definition
0	Unknown
1	Out of resource
2	Temporary situation
3	Authorization failure
4	Internal error
5	Hardware failure

Alert_class	NetWare Definition
6	System failure
7	Request error
8	Not found
9	Bad format
10	Locked
11	Media failure
12	Item exists
13	Station failure
14	Limit exceeded
15	Configuration error
16	Limit almost exceeded
17	Security audit information
18	Disk information
19	General information
20	File compression
21	Protection violation

The following NetWare events are defined in the BAROC file:

Event Class			Default Severity
NW4_Base			UNKNOWN
	NW4_SysLog_Base		UNKNOWN
		NW4_ClassUnknown	UNKNOWN
		NW4_OutOfResource	UNKNOWN
		NW4_TempSituation	UNKNOWN
		NW4_AuthorizationFailure	UNKNOWN
		NW4_InternalError	UNKNOWN
		NW4_HardwareFailure	UNKNOWN
		NW4_SystemFailure	UNKNOWN
		NW4_RequestError	UNKNOWN
		NW4_NotFound	UNKNOWN
		NW4_BadFormat	UNKNOWN
		NW4_Locked	UNKNOWN
		NW4_MediaFailure	UNKNOWN
		NW4_ItemExists	UNKNOWN
		NW4_StationFailure	UNKNOWN
		NW4_LimitExceeded	UNKNOWN
		NW4_ConfigurationError	UNKNOWN
		NW4_LimitAlmostExceeded	UNKNOWN
		NW4_SecurityAuditInfo	UNKNOWN
		NW4_DiskInformation	UNKNOWN

Event Class			Default Severity
		NW4_GeneralInformation	UNKNOWN
		NW4_FileCompression	UNKNOWN
		NW4_ProtectionViolation	UNKNOWN
	NW4_AppMessage		UNKNOWN
	NW4_NLM_Loading		UNKNOWN
	NW4_NLM_Unloaded		UNKNOWN
	NW4_NLM_NotLoaded		UNKNOWN
	NW4_Abend		UNKNOWN

TECADNW4.NLM

The NLM, **tecadnw4.nlm**, is the NetWare log file adapter. The commands for loading and unloading the NLM are described on the following pages.

tecadnw4.nlm

Starts the NetWare log file adapter in non-service mode.

Flags

Load tecadnw4 [**–c** *ConfigFile*] [**–d**]

Description

Loading **tecadnw4.nlm** starts the adapter. To stop the adapter, run the following from the command line:

```
unload tecadnw4
```

Authorization: None is required.

Arguments:

–c *ConfigFile*
: Specifies the configuration file for the NetWare log file adapter. If a value is not specified, the **TECADNW4.CNF** file in the current directory is used. If the **–c** argument is used, you can optionally specify a full path name for the configuration file; otherwise, the default configuration file, **SYS:ETC\TIVOLI\TECAD\ETC\ TECADNW4.CNF**, is used.

–d
: Shows verbose diagnostic information in the NLM screen as events are gathered and transmitted. Press the **Alt+Esc** or **Ctl+Esc** keys to switch to other NLMs screens or to return to the console.

  Note: Without the **–d** option, the adapter displays the initial startup messages on its screen but will close it upon completion of initialization, and the adapter name will not be displayed in the list of NLMs when the **Ctrl+Esc** keys are pressed.

Examples

The following command starts the NetWare log file adapter in debug mode:

```
load tecadnw4 -d
```

The following command starts the NetWare log file adapter with the **myconf.cnf** configuration file:

```
load tecadnw4 -c sys:etc\tmp\myconf.cnf
```

Troubleshooting the NetWare Log File Adapter

Perform the following steps to troubleshoot the NetWare log file adapter:

1. Stop the NetWare log file adapter that is currently running by unloading **tecadnw4.nlm**:

   ```
   unload tecadnw4
   ```

2. Start the adapter in debug mode:

   ```
   load tecadnw4 -d -c Config_File
   ```

3. Generate some events and see if the adapter receives them.

   As events arrive, the adapter prints messages to the screen indicating the class and the attribute values in the class.

4. As messages are displayed, run the **wtdumprl** command on the event server and verify that the messages are displayed or saved in the reception log. If not, the events were not received by the event server or there is a problem with the event server reception process.

5. Check the adapter configuration file to verify that **ServerLocation** and **ServerPort** are properly defined. If the event class appears in any filter entry in the configuration file, and **FilterMode=OUT**, the event is not sent to the event server.

6. If the reception log has a PARSING_FAILED error, the BAROC definition of the class does not match the event that is being received from the adapter. Usually the error messages pinpoint the problem.

7. If the previous steps do not indicate any problems and you do not see the new events in the event console, there might be a problem with the event group filters. Make sure the class filters match the classes defined in the BAROC files.

8. Change all **/dev/null** entries in the **.err** file to the file name you want. Stop and restart the adapter, send an event through, and then look in the trace file to see what processing was done on the event.

Chapter 5. OpenView Adapter

The IBM Tivoli Enterprise Console adapter for the Hewlett-Packard OpenView (HPOV) product forwards events from OpenView to the event server. The adapter is registered with the startup configuration of the OpenView operating system using **ovaddobj**, so it is started along with all the other applications that use the operating system. The OpenView **ovspmd** process manages the adapter and forwards all preferred events to the event server.

This chapter explains how to configure and start the OpenView adapter.

OpenView Driver

The OpenView adapter collects OpenView trap messages that have been sent by OpenView trap daemon (**ovtrapd**) and processed by the **ovspmd** daemon. The adapter translates the trap messages into the appropriate IBM Tivoli Enterprise Console class based on the entry that the trap matches in the **.cds** file.

Reception of OpenView Messages

In order to receive events generated by the OpenView Network Node Manager (NNM) and any events from all possible OpenView agents, the OpenView adapter registers itself into the NNM **SUF** startup file using the **ovaddobj** command. The **ovspmd** daemon reads **SUF** at startup and manages all the registered processes it finds, then receives events from the **ovtrapd** process and forwards the specified events to the appropriate registered applications (such as the OpenView adapter).

The OpenView adapter must run as a well-behaved daemon process using the OVsPMD API (application programming interface) functions provided with OpenView. The OVsPMD API functions are used by object managers (agents) that must run as background processes in the OpenView program in order to be managed by **ovspmd**, the process management daemon. The adapter interacts with **ovspmd** using the SNMP API functions provided with OpenView NNM. This involves the following steps:

- In NNM 5, calling **OVsnmpTrapOpen** to establish a logical session with the **OVsnmpAPI** to receive SNMP events through the OpenView Event Framework.
- In NNM 6, calling **OVsnmpEventOpen** to establish a logical session with the **OVsnmpAPI** to receive SNMP events through the OpenView Event Framework.
- Calling **OVsinit** to get a socket for communication with the **ovspmd** process.
- Calling **OVslnitComplete** to notify at the end of the initialization, the status of the initialization process.
- Calling **OVsReceive** to receive commands from the **ovspmd** process.
- Calling **OVsDone** to notify **ovspmd** that the adapter is being shut down.

Determining the OpenView NNM Version

To determine which version of OpenView NNM you are running, use the following command:

```
$OV_BIN/ovnnmversion
```

Incoming Messages Format

Messages received from the **ovtrapd** process consist of SNMP Trap-PDUs as defined in RFC 1157 (SNMPv l).

OpenView-specific events are defined as enterprise-specific traps and have the following content:

enterprise
1.3.6.1.4.1.11.2.17 for OpenView events

agent-addr
SNMP agent or proxy agent address

generic-trap
6

specific-trap
Number in the range 33554432 through 2147483647

time-stamp
0

variable-bindings
The adapter also receives SNMP traps because the **ovtrapd** process is monitoring for any traps sent to port 162. The following list shows some of the specifics for OpenView events:

1.	Descr:	OpenView Source ID number
	ObjId:	1.3.6.1.4.1.11.2.17.2.1.0
	Type:	INTEGER
2.	Descr:	OpenView Source Name
	ObjId:	1.3.6.1.4.1.11.2.17.2.2.0
	Type:	OCTET_STRING
3.	Descr:	OpenView Optional Object Id for event source
	ObjId:	1.3.6.1.4.1.11.2.17.2.3.0
	Type:	OCTET_STRING
4.	Descr:	Optional data
	ObjId:	1.3.6.1.4.1.11.2.17.2.4.0
	Type:	OCTET_STRING
5.	Descr:	Optional severity
	ObjId:	1.3.6.1.4.l.11.2.17.2.5.0
	Type:	OCTET_STRING
6.	Descr:	Optional category
	ObjId:	1.3.6 1.4.1.11.2.17.2.6.0
	Type:	OCTET_STRING

Event Correlation With NNM 6

You can configure the adapter to open a session with **ovspmd** so that **ovspmd** only forwards the correlated events you want to the adapter. This reduces the workload

on the adapter in proportion to the number of events discarded by the NNM circuit settings and therefore not forwarded to the adapter. If you are running NNM 5 or earlier, the adapter calls **OVsnmpTrapOpen** to open a session; with NNM 6 or later, the adapter calls **OVsnmpEventOpen**. Only **OVsnmpEventOpen** allows for event correlation of the events before they are forwarded to the adapter.

OVsnmpEventOpen contains a **filter** parameter that defines which events the application receives from **ovspmd**. A **filter** value of NULL or the empty string (“”) prevents the adapter from receiving any events and makes the session a send-only session; therefore, this is not a recommended configuration. See the manual page for **OVsnmpEventOpen** for more information.

The configuration file keyword **HPOVFilter** passes the filter value you specify to **OVsnmpEventOpen**. **HPOVFilter** specifies what kind of events are forwarded to the adapter from **ovspmd** and contains the value that will be used for the **filter** parameter when calling the **OVsnmpEventOpen** API. If you have NNM 6 and **HPOVFilter** is not specified or is commented out, the adapter receives all events by default. For more information about **HPOVFilter**, see “Configuration File” on page 70.

Determining the OVsnmpEventOpen Filter Value

The following examples show two ways to see how the value in **HPOVFilter** is passed to **OVsnmpEventOpen**.

- **Example 1**: NNM input event tracing is turned on and adapter tracing is turned off.

  Look in the file **$OV_LOG/ecs/<ecs-instance#>/ecsin.evt#** and do a find on previous `tecad_hpov` from the bottom of the file. The following example is similar to what you can see (the filter in this example is {CORR{default}} .*):

```
Trap-PDU {
    enterprise {1 3 6 1 4 1 11 2 17 1},
    agent-addr internet : "\x92T$\057",
    generic-trap 6,
    specific-trap 59179056,
    time-stamp 0,
    variable-bindings {
        {
            name {1 3 6 1 4 1 11 2 17 2 1 0},
            value simple : number : 14
        },
        {
            name {1 3 6 1 4 1 11 2 17 2 7 0},
            value simple : string : \             "{CORR{default}} .*"
        },
        {
            name {1 3 6 1 4 1 11 2 17 2 9 0},
            value application-wide : address : internet : "\x92T$\057"
        },
        {
            name {1 3 6 1 4 1 11 2 17 2 8 0},
            value simple : string : "tecad_hpov"
        },
        {
            name {1 3 6 1 4 1 11 2 17 2 10 0},
            value simple : number : 14128
        }
    }
}
% ber:Trap-PDU:
```

- **Example 2:** Adapter tracing is turned on by specifying output files in the **.err** file instead of **/dev/null**.

  You can find the NNM version and the specified filter value in the messages displayed when you start the adapter. The messages are similar to the following example:

```
Initializing T/EC interface ...
T/EC interface initialization complete
Initializing driver ...
Initializing SNMP driver ...
Running as a WellBehavedDaemon
Enter in TECAD_OVsInit...
HP NNM version running is: HP OpenView ov library \
NNM Release B.06.10 @(#) PATCH PSOV_XXXXX, YYMMDD Oct 17 1999
Stream filtering set to: {CORR{default}} .*
```

Testing Tools

In order to test the OpenView adapter, it is necessary to have OpenView installed on the same system on which the adapter is running. Testing of the adapter behavior can only be achieved by starting all daemon processes of OpenView and by sending SNMP trap events to the **ovtrapd** process. Note that SNMP trap events can be generated by sending SNMP traps to **ovtrapd** using the same testing tool as for the SNMP adapter.

With OpenView, it is also possible to simulate events occurring by using **smnptrap(1)**, **ovevent**, or by using specific commands such as:

- **OV_Set_status_Color** (specific trap number 58916871)
- **OV_Message** (specific trap number 58916872)
- **OV_Popup_Message** (specific trap number 58916873)
- **OV_Bell_Message** (specific trap number 58916874)
- **OV_Highlight_Source** (specific trap number 58916875)

An example using **snmptrap(1)** for creating a message and ringing a bell from node **Bad_Node** is presented as follows:

```
snmptrap 'hostname' \
  1.3.6.1.4.1.11.2.17.1 ""6 58916874"" \
  1.3.6.1.4.1.11.2.17.2.1.0 Integer 14 \
  1.3.6.1.4.1.11.2.17.2.2.0 OctetString "Bad_Node" \
  1.3.6.1.4.1.11.2.17.2.4.0 OctetString "Bell Message"
```

Testing Event Correlation With NNM 6

Stream and circuit tracing can help you see which events will be forwarded to the adapter. A stream with an output policy forwards any event unless you enable at least one circuit on the stream to discard a type of event. A stream with a discard policy only forwards an event if you enable a circuit on the stream that outputs that type of event. An output file lists the forwarded events. For example, when a stream has an output policy, you can determine what events that the stream sent to the adapter by reading the events listed in the stream output file.

For complete details on streams and circuits, see the HP OpenView NNM documentation.

The following lists show some of the commands you can use with streams and circuits:

- To find details about the event correlation engine, use the following command:
  `ecsmgr -info`

- To find details about event arrivals for the circuits and streams, use the following command: `ecsmgr -stats`
- To turn on tracing to see the OpenView events received, use the following command: `ecsmgr -log_events input on`

  This trace file is located in **$OV_LOG/ecs/<ecs-instance#>/ecsin.evt#**
- To turn on tracing to see the OpenView stream events received, use the following command:

  `ecsmgr -log_events stream <stream-name> on`

  The trace files for the stream output events are located in **$OV_LOG/ecs/<ecs-instance#>/<*stream-name*>_sout.evt#**

  The trace files for the discarded stream events are located in **$OV_LOG/ecs/<ecs-instance#>/<*stream-name*>_sdis.evt#**

  The following example turns on stream event tracing for a stream named **default**:

  `ecsmgr -log_events stream default on`
- To turn on tracing to see the OpenView circuit events received, use the following command:

  `ecsmgr -log_events circuit <circuit-name> on`

  The trace files for the circuit output events are located in **$OV_LOG/ecs/<ecs-instance#>/<*circuit-name*>_cout.evt#**

  The trace files for the discarded circuit events are located in **$OV_LOG/ecs/<ecs-instance#>/<*circuit-name*>_cdis.evt#**

  The following example turns on circuit event tracing for a stream named **PairWise**:

  `ecsmgr -log_events circuit PairWise on`

Event Correlation Example

The following event passes through circuits named PairWise and ConnectorDown. When the **HPOVFilter** value passed to **OVsnmpEventOpen** is `.*`, the event is forwarded to the adapter because the stream **default** is not being used. If the **HPOVFilter** value is `{CORR{default}} .*`, you can only see the event in the circuit discard trace file.

```
snmptrap <boxname> "1.3.6.1.4.1.11.2.17.1" 146.84.36.175 6 40000084 0 \
1.3.6.1.4.1.11.2.17.2.1.0 integer 7 \
1.3.6.1.4.1.11.2.17.2.2.0 octetstringascii "snmp trap for connector down"
```

Note: You must watch the circuit and stream trace files to see when this event is discarded. This event sometimes is sent to the adapter instead. Keep the message text changing slightly so that you can identify a specific event. Also, send multiple events until the discard trace file for the stream **default** shows the event is discarded, which indicates that the event was not sent to the adapter.

The following event is sent to the adapter when **HPOVFilter** is set to `{CORR{default}} .*`:

```
/opt/OV/bin/ovevent -s Major -c "Error Events" "" \
          .1.3.6.1.4.1.11.2.17.1.0.58916872 \
           .1.3.6.1.4.1.11.2.17.2.1.0 Integer 14 \
           .1.3.6.1.4.1.11.2.17.2.2.0 OctetString "user@host" \
           .1.3.6.1.4.1.11.2.17.2.4.0 OctetString "major error message"
```

Adapter Files

The OpenView adapter package consists of the following files in the following directories:

- **$TECADHOME/bin**

  tecad_hpov.cfg
  The installation configuration script.

  tecad_hpov
  The adapter executable file.

  tecad_hpov.sh
  The adapter shell script to set the environment and call the adapter executable file.

- **$TECADHOME/etc**

  tecad_hpov.baroc
  The adapter BAROC file to define the classes to the rule base.

  tecad_ov.baroc
  An additional BAROC file that precedes **tecad_hpov.baroc** in the rulebase definitions to define the enumerations that **tecad_hpov.baroc** uses.

  tecad_hpov.cds
  The class definition statement (CDS) file. This file defines the adapter class definitions.

  tecad_hpov.conf
  The configuration file. This file defines the adapter startup configuration.

  tecad_hpov.err
  The error file. This file indicates where to write adapter trace messages.

  tecad_hpov.lrf
  The registration file. This file is generated by the installation configuration script and placed in the **$OV_LRF** directory. For UNIX, the directory is usually **/etc/opt/OV/share/lrf**. For Microsoft Windows NT, the directory is usually **c:/Openview/LRF/tecad_hpov.lrf**.

  tecad_hpov.oid
  The object identifier file. This file matches object identifiers to variable names.

  ov_default.rls
  The default rule file for the OpenView adapter used in the rule base.

Before starting the adapter, check each adapter file to ensure that they define the preferred adapter behavior.

Configuration File

The configuration file of the OpenView adapter defines the behavior of the adapter, which runs as a server daemon. The configuration file can have common keywords described in “Configuration File” on page 9, as well as the following adapter-specific keywords:

AdapterSpecificFile=*path*
Specifies the full path name of the object identifier file. This keyword is required if the object identifier file is not in the same directory as the configuration file.

HPOVFilter=*filter*

Specifies the events the adapter receives from OpenView NNM 6. This value is ignored with OpenView NNM 5. The adapter can accept up to 4096 bytes for this parameter; you must enter the value in one continuous line of input with no intervening line returns. Do not enclose the value in quotation marks; if you enclose the value in quotation marks and turn on adapter tracing, the trace file displays the following error:

```
Stream filtering set to: "{CORR{default}} .*"
Enter in TECAD_OVsInit...
Unable to initialize SNMP session system error: Invalid event \
filter (Filter parameter (""{CORR{default}}.*"") event \
specification must be "" or start with a '.')
Unable to initialize SNMP session system error: Bad file number
Enter in TECAD_OVsInitComplete...
can not initialize specific driver
```

The adapter also fails to initialize, and **ovspmd** sends the following message:

```
# ovstart tecad_hpov
object manager name: tecad_hpov
state:               FAILED
PID:                 12901
last message:        Unable to initialize SNMP session
system \
error:      Bad file number
exit status:         -
```

Turn on adapter tracing when you change the value for **HPOVFilter** to make sure that the value was entered correctly or to see the errors generated by it.

See the manual page for **OVsnmpEventOpen** for details on **HPOVFilter** and the **filter** parameter.

WellBehavedDaemon

Specifies whether the adapter runs as an OpenView well-behaved daemon. This value should always be **TRUE**.

Class Definition Statement File

The CDS file defines how events are constructed from the information that is sent by OpenView. It is described in detail in “Class Definition Statement File” on page 18 and in Appendix C, “Class Definition Statement File Reference” on page 155.

Errors in the **.cds** file definitions cause the adapter to not start successfully, which often causes the adapter to exit with an `exit (1)`. Therefore, change one definition at a time and restart the adapter after each change to ensure that the new definition works. If you make many changes before restarting the adapter, it is more difficult to troubleshoot any problems; turning on adapter tracing helps you locate the errors.

OpenView Event Example

The class definition in the following example is taken from the **.cds** file:

```
CLASS_OV_IF_FAULT
 SELECT
  1:ATTR(=,ENTERPRISE),VALUE(PREFIX, \
   "1.3.6.1.4.1.11.2.17.1");
  2:$SPECIFIC=40000000;
  3:ATTR(=, "openViewSourceName");
```

```
  4:ATTR(=, 'openViewData3");
  5:ATTR(=, "openViewData4");
 MAP
  origin=$V3;
  sub_origin=$V4;
  severity=WARNING;
  OV_status=2; # Marginal
```

Keywords

The OpenView adapter supports the use of the following keywords in class definition statements. These keywords can be useful if you want to customize events.

$COMMUNITY
: Specifies the trap community string.

$ENTERPRISE
: Specifies the enterprise object identifier of the object generating the trap.

$SOURCE_TIME
: Specifies the value of **sysUpTime** of the object generating the trap.

$TYPE Specifies the generic trap type number (0-6).

$SPECIFIC Specifies the enterprise-specific trap type number.

$SOURCE_ADDR
: Specifies the address of the object sending the trap.

$AGENT_ADDR
: Specifies the address of the object generating the trap.

$VARBIND Specifies a list of all non-fixed attributes.

$VB_NUM_VARS
: Specifies the number of elements in **$VARBIND**.

$ADAPTER_HOST
: The name of the host machine where the adapter runs.

The following example shows how you can use the keywords:

```
FETCH
    1: IPNAME($SOURCE_ADDR);

SELECT
    1: ATTR(=, $ENTERPRISE);
```

Built-in Variables for $VARBIND: **$VARBIND** is a list of all non-fixed attributes. To access the individual elements of **$VARBIND**, use the **VB_#** variables, where **#** is a number greater than 0. For example, if **$VARBIND** has three elements, you can use **VB_1**, **VB_2**, and **VB_3** as variables to access the data. The following example performs string functions on the elements of **$VARBIND**.

```
ATTR(=, "VB_1"), VALUE(CONTAINS, "some string")
```

Because **$VARBIND** is a list of strings, if it contains more than one element, performing a string function like CONTAINS against **$VARBIND** causes the adapter to stop unexpectedly.

Object Identifier File

The object identifier file maps object identifiers used by SNMP to names. No changes are necessary before the adapter is run.

Each line of this file has the following form:

```
"name"   "object identifier"
```

For example

```
"sysUpTime"   "1.3.6.1.2.1.1.3"
"ifIndex"   "1.3.6.1.2.1.2.2.1.1"
"whyReload"   "1.3.6.1.4.1.9.2.1.2"
```

Note: Object identifiers must appear in increasing order.

You can use the names that are mapped to object identifiers in the CDS file.

Error File

The error file enables you to configure debugging and tracing options. This file is described in detail in “Error File” on page 19.

LRF File

The **.lrf** file registers the application when the NNM application starts up. The **.lrf** file is created and registered automatically when the adapter is installed. For details on the syntax of the file, see the OpenView NNM documentation.

If you need to make changes to the **tecad_hpov.lrf** file, follow these steps:

1. Stop the adapter.
2. Change the **.lrf** file as needed and save it.
3. Register the change with NNM by using **$OV_BIN/ovaddobj $OV_LRF/tecad_hpov.lrf**.
4. Restart the adapter.

If the **tecad_hpov.lrf** file has errors, the adapter might not start successfully.

Starting and Stopping the Adapter

If you have configured the host start-up file correctly, the adapter always starts when the OpenView operating system starts up. You can also start an adapter manually. When the adapter starts up, it gets new bindings, reads its adapter files, and restarts the daemon.

Use the following commands to start and stop the adapter. You can access the OpenView NNM environment variables by sourcing the NNM environment using the **ov.envvars.sh** file in the **/bin** directory in the OpenView NNM installation directory.

```
. /opt/OV/bin/ov.evvars.sh     # source the unix/bash environment
/opt/OV/bin/ov.envvars.bat    # source the MS-DOS environment
$OV_BIN/ovstop tecad_hpov     # stop the OpenView adapter
$OV_BIN/ovstart tecad_hpov    # start the OpenView adapter
```

Events Listing

The following table shows the class names and severities of all events defined for the OpenView adapter. You can use it to get a sense of how OpenView events are mapped to IBM Tivoli Enterprise Console events and to determine if you want to make any changes. The events are defined in the BAROC file. See the *IBM Tivoli Enterprise Console Rule Builder's Guide* for more information about customizing the BAROC file.

Event Class Structure

Event classes are defined hierarchically, with child classes inheriting attribute value defaults from the parent. The OpenView event classes follow a simple hierarchy.

The adapter fills in the following attribute defaults. The attributes are used in event group filters.

source
HPOV

sub_source
NET

origin
hostIPaddress where the event originated

hostname
hostname where the event originated

adapter_host
Host on which the adapter runs

forwarding_agent
Proxy agent that forwarded the event to the adapter

Additional information is provided where possible by using OpenView category and status codes. See the **ENUMERATION** statements at the beginning of the BAROC file for details.

The following table shows events defined in the BAROC file.

Event Class		Default Severity
OV_Event		WARNING
	OV_Bad_Subnet_Mask	WARNING
	OV_CMIS_Event	WARNING
	OV_Change_Polling_Period	WARNING
	OV_Chg_IF_Segment	WARNING
	OV_Connection_Added	WARNING
	OV_Connection_Deleted	WARNING
	OV_DataCollectThresh	WARNING
	OV_DataCollect_Rearm	HARMLESS
	OV_Error	WARNING
	OV_Fatal_Error	FATAL
	OV_Forw_Status_Chg	MINOR
	OV_IF_Added	WARNING

Event Class			Default Severity
	OV_IF_Deleted		WARNING
	OV_IF_Descr_Chg		MINOR
	OV_IF_Fault		WARNING
	OV_IF_Down		FATAL
	OV_IF_Flags_Chg		WARNING
	OV_IF_Type_Change		MINOR
	OV_Manage_IF		WARNING
	OV_Manage_Network		WARNING
	OV_Manage_Node		WARNING
	OV_Manage_Segment		WARNING
	OV_Network_Added		HARMLESS
	OV_Network_Deleted		WARNING
	OV_Network_Fault		WARNING
		OV_Network_Critical	CRITICAL
		OV_Network_Marginal	WARNING
		OV_Network_Normal	HARMLESS
	OV_Network_Flg_Chg		WARNING
	OV_No_SNMP_Reply		CRITICAL
	OV_Node_Added		WARNING
	OV_Node_Deleted		WARNING
	OV_Node_Fault		FATAL
		OV_Node_Down	WARNING
		OV_Node_Marginal	WARNING
	OV_Node_Flags_Chg		WARNING
	OV_Object_ID_Chg		MINOR
	OV_Phys_Addr_Chg		MINOR
	OV_Phys_Addr_Mismatch		MINOR
	OV_Segment_Added		HARMLESS
	OV_Segment_Deleted		WARNING
	OV_Segment_Fault		WARNING
		OV_Segment_Critical	CRITICAL
		OV_Segment_Marginal	WARNING
		OV_Segment_Normal	HARMLESS
	OV_Segment_Flag_Chg		WARNING
	OV_Subnet_Mask_Chg		MINOR
	OV_Sys_Contact_Chg		HARMLESS
	OV_Sys_Descr_Chg		HARMLESS
	OV_Sys_Location_Chg		HARMLESS
	OV_Sys_Name_Chg		HARMLESS
	OV_Unmanage_IF		WARNING
	OV_Unmanage_Network		WARNING

Event Class		Default Severity
	OV_Unmanage_Node	WARNING
	OV_Unmanage_Segment	WARNING
HPOV_Event		WARNING
	OV_ARP_Chg_New_Phys_Addr	WARNING
	OV_ARP_Phys_Chg_Same_Src	WARNING
	OV_AppUngracefulExit	WARNING
	OV_Application_Alert	WARNING
	OV_Application_Down	WARNING
	OV_Application_Up	WARNING
	OV_Bad_Forw_To_Host	WARNING
	OV_Bad_Phys_Address	WARNING
	OV_ConnectionUnknown	WARNING
	OV_Connection_Down	FATAL
	OV_DataCollect_Check	WARNING
	OV_IF_Disconnected	WARNING
	OV_IF_IP_Addr_Chg	WARNING
	OV_IF_Unknown	WARNING
	OV_Map_Change	WARNING
	OV_Network_IPAddrChg	WARNING
	OV_Network_Name_Chg	WARNING
	OV_Network_SubMskChg	WARNING
	OV_Network_Unknown	WARNING
	OV_Node_SupportsSNMP	WARNING
	OV_Node_Unknown	WARNING
	OV_Segment_Unknown	WARNING
	OV_Trap_PDU_Error	WARNING

OpenView Traps

SNMP Traps

All SNMP generic traps and enterprise-specific traps supported by the SNMP adapter are also supported by the OpenView adapter.

OpenView Traps

OpenView events are SNMP traps, and their content has been described within “OpenView Driver” on page 65.

The specific-trap is the number identifying the sub-type of the trap. For OpenView events, the following list is used:

50462720 Warnings

50790400 Node Marginal

50790401 Segment Normal

50790402	Segment Marginal
50790403	Network Normal
50790404	Network Marginal
50790405	Segment Added
50790406	Segment Deleted
50790407	Network Added
50790408	Network Deleted
50790409	Connection Added
50790410	Connection Deleted
50790411	Change Polling Period
50790412	Forced Poll
50790418	Manage Node
50790419	Unmanage Node
50790420	Manage Segment
50790421	Unmanage Segment

All OpenView events are supported by the OpenView adapter.

Troubleshooting the OpenView Adapter

Perform the following steps to troubleshoot the OpenView adapter:

1. Make sure that the **tecad_hpov.lrf** entry is correct and has been registered with OpenView using the **ovaddobj** command.
2. If the adapter does not start, look for errors in the **.lrf**, **.oid,** and **.cds** files.
3. If the adapter stops unexpectedly, look for data that is not valid being passed in a trap or functions. For example, **PREFIX** is called on a list of strings value instead of a string value.
4. Change all **/dev/null** entries in the **.err** file to the file name you want. Stop and restart the adapter, send an event through, and then look in the trace file to see what processing was done on the event.
5. Look in **/tmp/hpov_start.err** for possible startup errors from the **tecad_hpov.sh** script.

Chapter 6. OS/2 Adapter

The IBM Tivoli Enterprise Console adapter for OS/2 forwards events from an OS/2 system to the event server. The adapter is registered with the startup configuration of OS/2 so that the adapter is started with all the other applications that are automatically started when OS/2 is started.

The adapter is an OS/2 process that reads events generated by an OS/2 system and forwards them to an event server for further processing.

OS/2 events are gathered from the First Failure Support Technology™ (FFST™) system, and from ASCII log files residing on the OS/2 system. The adapter translates a certain type of FFST events into IBM Tivoli Enterprise Console events and sends them to the event server. There are three types of FFST events: DET1, DET2, and DET4. DET1 events represent error conditions and are the only type sent to the event server. Entries in the ASCII log files are formatted according to the format file.

This chapter describes how to configure and start the OS/2 adapter.

Adapter Files

The OS/2 adapter package consists of the following files:

readme The readme file.

tecadcfg.cmd The startup configuration script.

tecadini.sh The script to start or stop the adapter.

tecadrm.sh The TME adapter uninstall script.

tec_uninstal.cmd
The non-TME adapter uninstall batch file.

install.exe The adapter installation assist executable file.

tecados2.exe The adapter executable.

tecados2.conf The configuration file.

tecados2.fmt The format file.

tecados2.cds The class definition statement (CDS) file.

tecados2.baroc The BAROC file.

tecados2.err The error file.

Configuration File

The configuration file defines the behavior of the adapter. This file can contain the common keywords described in “Configuration File” on page 9, as well as the following adapter-specific keywords:

LogSources
Specifies the ASCII log files to monitor for messages. The complete path to each file must be specified, and file names must be separated by commas; no spaces or other separators can be used. A log file source need not exist when the adapter is started; it will be monitored when it is created.

If a file truncates while the adapter is active, the adapter automatically resets its internal pointer to the beginning of the file. If during the polling interval the file is overwritten, removed, or recreated with more lines than the previous poll, only the number of lines greater than the previous line count is read. For example, the file has one line. After the poll interval elapses, the file is overwritten with two lines. Only the second line is read on the next polling.

UnmatchLog
Specifies a file to log discarded events that cannot be parsed into an IBM Tivoli Enterprise Console event class by the adapter. The discarded events can then be analyzed to determine if modifications are needed to the adapter format file.

Format File

The format file contains message format descriptions and their mapping to BAROC events. The message fields of an OS/2 event are matched against the format descriptions in this file and when a match succeeds, the corresponding IBM Tivoli Enterprise Console event is generated by the adapter. The format file contains predefined mappings for some common OS/2 events and can be customized to add any new messages.

The OS/2 adapter extracts the following information from an FFST event:

- Date of the event
- Name of the host that issued the event
- Process name associated with the event
- Severity of the event
- Probe ID
- Module name
- Message text

For details about format files, see “Format File” on page 17 and Appendix B, “Format File Reference” on page 145.

Starting the Adapter

By default, the adapter is started when OS/2 is started. To manually start the adapter, perform the following steps from the OS/2 desktop:

1. Open the **System** folder.
2. Open the **Startup** folder.
3. Double-click the **TEC Adapter** icon.

Note: The endpoint version of the adapter is started when the adapter configuration profile (ACP) is distributed using the Adapter Configuration Facility (ACF). Non-TME adapters are started during adapter installation.

You can also manually start the adapter by entering the following command sequence from the OS/2 command line:

```
sh %LCF_BINDER%/../TME/TEC/ADAPTERS/BIN/tecadini.sh start
```

Stopping the Adapter

You can manually stop the endpoint adapter by sourcing the endpoint environment, and then entering the following command sequence from the OS/2 command line:

```
sh %LCF_BINDIR%/../TME/TEC/ADAPTERS/BIN/tecadini.sh stop
```

You can manually stop the non-TME adapter from the OS/2 command line with the following command sequence:

```
%INSTALL_DIR%\BIN\WOS2KILL.EXE -a
```

Events Listing

The following table shows the class names and severities of all events defined for the OS/2 adapter. You can use it to get a sense of how OS/2 events are mapped to IBM Tivoli Enterprise Console events and to determine if you want to make any changes. The events are defined in the BAROC file.

See the *IBM Tivoli Enterprise Console Rule Builder's Guide* for more information about customizing a BAROC file.

Event Class Structure

Event classes are defined hierarchically, with child classes inheriting attribute value defaults from the parent. The OS/2 event classes follow a simple hierarchy.

The adapter fills in the following attribute default values. The attributes are used in event group filters.

source OS2

sub_source
OS2

The following events are defined in the BAROC file:

Event Class	Default Severity
OS2_Base	4 (WARNING)
OS2_FFST_Base	4 (WARNING)

The severity is set using numeric values in the format file, which you can modify to set the severity of a specific message. The following table shows the numeric values and their literal values:

Numeric Value	Literal Value
1	FATAL
2	CRITICAL
3	MINOR
4	WARNING
5	UNKNOWN

Numeric Value	Literal Value
6	HARMLESS

Troubleshooting the OS/2 Adapter

Perform the following steps to troubleshoot the OS/2 adapter:

1. Stop the OS/2 adapter that is currently running. See “Stopping the Adapter” on page 81 for details.
2. Add a **LogSources=c:\check.txt** entry in the configuration file.
3. Start the adapter as described in “Starting the Adapter” on page 80.
4. Add a few lines to **c:\check.txt**.
5. Run the **wtdumprl** command on the event server and verify that the messages are actually showing up in the reception log. If not, the events were not received by the event server or there is a problem with the event server reception process. Check the adapter configuration file to verify that **ServerLocation** and **ServerPort** are properly defined. If the event class appears in any filter entry in the configuration file, the event is not sent to the server. The administrator who started the adapter must have the required roles if running the TME version of the adapter. For a TME adapter, running the **odstat** command can offer some clues as to what could have failed.
6. If the reception log has a **PARSING_FAILED** error, the BAROC definition of the class does not match the event that is being received from the adapter. Usually the error messages pinpoint the problem. If the previous steps do not indicate any problems and you do not see the new events in the IBM Tivoli Enterprise Console product, there might be a problem with the event group filters. Make sure the class filters match the classes in the BAROC file.
7. Change all **/dev/null** entries in the **.err** file to the file name you want. Stop and restart the adapter, send an event through, and then look in the trace file to see what processing was done on the event.

Chapter 7. SNMP Adapter

The Simple Network Management Protocol (SNMP) adapter for the IBM Tivoli Enterprise Console product forwards events from SNMP traps to the event server.

This chapter explains how to configure and start the SNMP adapter.

SNMP Driver

The SNMP adapter serves the function of collecting SNMP trap messages directly from the SNMP trap socket of a host and translating SNMP traps into appropriate IBM Tivoli Enterprise Console class instances.

The SNMP manipulation routines make use of SNMP Research SNMP libraries.

Reception of SNMP Messages

The SNMP adapter receives SNMP traps by listening directly on socket udp/162 of the host it runs on.

Incoming Messages Format

Messages received on the udp/162 socket consist only of SNMP Trap-PDUs as defined in RFC 1157 (SNMPv1). Other types of messages are discarded.

Server Configuration

Since the SNMP trap adapter listens on UDP socket 162 for incoming SNMP traps, it must be run as **root**. Also, UDP socket 162 must not already be in use by another SNMP manager, such as the **trapd** daemon for IBM NetView for AIX® or the SNMP trap daemon itself.

Adapter Files

The SNMP adapter package consists of the following files:

tecad_snmp.cfg
The installation script.

tecad_snmp The adapter executable file.

tecad_snmp.baroc
The BAROC file.

tecad_snmp.cds
The class definition statement (CDS) file.

tecad_snmp.conf
The configuration file.

tecad_snmp.err
The error file.

tecad_snmp.oid
The object identifier file.

init.tecad_snmp
The adapter startup and shutdown script.

Before starting the adapter, check each adapter file to determine if it defines the behavior you want from the adapter.

Configuration File

The configuration file defines the behavior of the adapter, which runs as a server daemon. The configuration file can have the common keywords described in "Configuration File" on page 9, as well as the following adapter-specific keywords:

AdapterSpecificFile=*path*
: Specifies the full path name of the object identifier file. This keyword is required if the object identifier file is not in the same directory as the configuration file**.**

SNMP_PORT Specifies the port where the adapter listens for SNMP requests**.**

SNMP_TRAP Specifies the port where the adapter listens for SNMP traps. Only change this value if the producers of events are configured to send to the alternate port**.**

Class Definition Statement File

The CDS file defines how events are constructed from information sent by SNMP. It is described in detail in "Class Definition Statement File" on page 18 and in Appendix C, "Class Definition Statement File Reference" on page 155.

SNMP Event Example

```
CLASS Port_Segmenting_CBT
SELECT
  1:ATTR(=,$ENTERPRISE),VALUE(PREFIX, "1.3.6.1.4.1.52") ;
  2:$SPECIFIC=258 ;
  3:ATTR(=,"boardIndex") ;
  4:ATTR(=,"portIndex") ;
FETCH
  1:IPNAME($SOURCE_ADDR) ;
MAP
  hostname=$F1 ;
  boardIndex=$V3 ;
  portIndex=$V4 ;
  sub_origin=PRINTF("board %s, port %s", $V3, $V4) ;
  status=CLOSED ;
END
```

Keywords

To customize events, use the following keywords in class definition statements. Event definition content and syntax are described in the *IBM Tivoli Enterprise Console Rule Builder's Guide*.

$COMMUNITY
: Specifies the trap community string.

$ENTERPRISE
: Specifies the enterprise object identifier of the object generating the trap.

$SOURCE_TIME
: Specifies the value of **sysUpTime** of the object generating the trap.

$TYPE Specifies the generic trap type number (0-6).

$SPECIFIC Specifies the enterprise-specific trap type number.

$SOURCE_ADDR
: Specifies the address of the object sending the trap.

$AGENT_ADDR
Specifies the address of the object generating the trap.

$VARBIND Specifies a list of all non-fixed attributes.

$VB_NUM_VARS
Specifies the number of elements in **$VARBIND**.

$ADAPTER_HOST
The name of the host machine where the adapter runs.

Built-in Variables for $VARBIND: **$VARBIND** is a list of all non-fixed attributes. To access the individual elements of **$VARBIND**, use the VB_# variables, where **#** is a number greater than zero (0). For example, if **$VARBIND** has three elements, you can use VB_1, VB_2, and VB_3 as variables to access the data. The following example performs string functions on the elements of **$VARBIND**:

```
ATTR(=, "VB_1"), VALUE(CONTAINS, "some string")
```

Because **$VARBIND** is a list of strings, if it contains more than one element, performing a string function like CONTAINS against **$VARBIND** causes the adapter to end unexpectedly.

Object Identifier File

The object identifier file maps object identifiers used by SNMP to names. No changes are necessary before the adapter is run.

Each line of this file has the following form:

```
"name"        "object identifier"
```

For example

```
"sysUpTime"        "1.3.6.1.2.1.1.3"
"ifIndex"          "1.3.6.1.2.1.2.2.1.1"
"whyReload"        "1.3.6.1.4.1.9.2.1.2"
```

Note: Object identifiers must appear in increasing order.

You can use the names that are mapped to object identifiers in the CDS file.

Error File

The error file allows you to configure debugging and tracing options. The error file is described in detail in “Error File” on page 19.

Starting and Stopping the Adapter

By default, the adapter is always started when the host starts up. You can also cold start or warm start an adapter manually. A cold start causes the adapter to get new bindings, read its adapter files, and restart the daemons. A warm start causes the server only to re-read its adapter files.

Unless explicitly defined in the configuration file, the adapter searches for the CDS, error, and object identifier files in the same directory as the configuration file.

Cold Start

The endpoint adapter is automatically started as a step in the adapter installation process when the adapter configuration profile (ACP) is distributed using the Adapter Configuration Facility (ACF).

Manually start the adapter on the endpoint with the following command:

```
init.tecad_snmp start
```

Warm Start

You can restart a running adapter. Doing so is useful when you have changed one of the adapter files and want to have it read in without bringing the adapter or host down completely.

Use one of the following **kill** commands to force the adapter to restart:

```
# kill -HUP process_number
```

—OR—

```
# kill -1 process_number
```

Stopping the Adapter

The endpoint adapter can be automatically stopped by distributing an ACP that has the adapter start command removed from the after-file-distribution actions. See the *IBM Tivoli Enterprise Console User's Guide* for additional information.

Manually stop the adapter on the endpoint with the following command:

```
init.tecad_snmp stop
```

Events Listing

The following table shows the class names and severities of all events defined for the SNMP adapter. You can use it to get a sense of how SNMP traps are mapped to IBM Tivoli Enterprise Console events and to determine if you want to make any changes. The events are defined in the BAROC file.

See the *IBM Tivoli Enterprise Console Rule Builder's Guide* for more information about customizing the BAROC file.

Event Class Structure

Event classes are defined hierarchically, with child classes inheriting attribute value defaults from the parent. The SNMP event classes follow a simple hierarchy.

The adapter fills in the following attribute defaults. The attributes are used in event group filters.

source
SNMP

sub_source
NET

origin
hostIPaddress where the event originated

hostname
hostname where the event originated

adapter_host
Host on which the adapter runs

forwarding_agent
Proxy agent that forwarded the event to the adapter

Additional information is provided where possible by using OpenView category and status codes. See the **ENUMERATION** statements at the beginning of the BAROC file for details.

The following events are examples of the ones defined in the BAROC file:

Event Class	**Event Severity**
SNMP_Trap	WARNING
Generic_SNMP_Trap	WARNING
Cold_Start	WARNING
Cold_Start_Cisco	WARNING
Warm_Start	WARNING
Link_Down	FATAL
Link_Down_Cisco	WARNING
Link_Up	HARMLESS
Authentication_Failure	WARNING
Authentication_Failure_Cisco	WARNING
EGP_Neighbor_Loss	CRITICAL
EGP_Neighbor_Loss_Cisco	WARNING
Specific_SNMP_Trap	WARNING
CBT_Trap	WARNING
Port_Segmenting_CBT	WARNING
Port_Link_Down_CBT	WARNING
Source_Address_New_CBT	WARNING
Source_Address_Timeout_CBT	WARNING
Board_Removal_CBT	WARNING
Board_Insertion_CBT	WARNING
Active_Port_In_Redundant_Circuit_Failed_ CBT	WARNING
Redundant_Port_Activated_CBT	WARNING
Redundant_Port_Test_Failed_CBT	WARNING
Device_Traffic_Threshold_Exceeded_CBT	WARNING
Device_Error_Threshold_Exceeded_CBT	WARNING
Device_Collision_Threshold_Exceeded_CBT	WARNING
Board_Traffic_Threshold_Exceeded_CBT	WARNING
Board_Error_Threshold_Exceeded_CBT	WARNING
Board_Collision_Threshold_Exceeded_CBT	WARNING
Port_Traffic_Threshold_Exceeded_CBT	WARNING
Port_Error_Threshold_Exceeded_CBT	WARNING
Port_Collision_Threshold_Exceeded_CBT	WARNING

Event Class				Event Severity
			Port_Type_Changed_CBT	WARNING
			Lock_Status_Changed_CBT	WARNING
			Port_Security_Violation_CBT	WARNING
			Port_Violation_Reset_CBT	WARNING
			Env_Temperature_CBT	WARNING
		Cisco_Trap		WARNING
			Reload_Cisco	WARNING
			TCP_Connection_Close_Cisco	HARMLESS

The **tecad_snmp.baroc** file contains a complete listing of events including NetWare, Cisco, Cabeltron, and generic traps. Refer to the BAROC file for details.

Rules Listing

There are no default rules for the SNMP adapter.

SNMP Traps

Generic Traps

All SNMP generic traps (**Cold_Start**, **Warm_Start**, **Link_Down**, **Link_Up**, **Authentication_Failure**, **Egp_Neighbor_Loss**) are mapped to distinct event classes.

These generic SNMP event classes can be specialized to incorporate additional information provided by some equipment. For instance, when a Cisco router issues an **Authentication_Failure** trap, it provides an additional variable in the varbind list that gives the protocol address of the device sending the badly authenticated SNMP request. For **Link_Down** traps, Cisco routers provide additional information describing which interface is going down and why it is going down. Since the content of the varbind list is not specified in the SNMP standard, it can vary from one equipment to the next. This can impact the way event classes and subclasses are defined.

Enterprise-specific Traps

By definition, enterprise-specific traps vary from one equipment vendor to the next.

Enterprise-specific traps can be handled by supporting Cisco routers enterprise-specific traps, as follows:

0 Reload

1 tcpConnectionClose

Additionally, enterprise-specific traps can be handled by supporting Cabletron hubs, as follows:

257 PortSegmenting

258 PortUnsegmenting

259 PortLinkUp

260 PortLinkDown

261 NewSourceAddress

262 SourceAddressTimeout

263 BoardRemoval

264 BoardInsertion

265 ActivePortInRedundantCircuitFailed

266 RedundantPortActivated

267 RedundantPortTesfFailed

268 DeviceTrafficThresholdExceeded

269 DeviceErrorThresholdExceeded

270 DeviceCollisionThresholdExceeded

271 BoardTrafficThresholdExceeded

272 BoardErrorThresholdExceeded

273 BoardCollisionThresholdExceeded

273 BoardCollisionThresholdExceeded

274 PortTrafficThresholdExceeded

275 PortErrorThresholdExceeded

276 PortCollisionThresholdExceeded

277 PortTypeChanged

278 LockSTATUSChanged

279 PortSecurityViolation

280 PortViolationReset

281 EnvTempWarm

282 EnvTempHot

283 EnvVoltageLow

Creating a New SNMP Trap Event

To create a new SNMP trap event using an SNMP Management Information Base (MIB) file, change the following files:

- **tecad_snmp.baroc**
- **tecad_snmp.cds**
- **tecad_snmp.oid**

This section describes traps from the LANAlert FSA for NetWare 3.*x*. Traps from other agents are similar.

BAROC File Changes

From this partial MIB file, create a **lanalertFSA-NW3-s1** event in the **tecad_snmp.baroc** file.

```
-- LANAlert Forwarding Gateway MIB (partial)
-- NCI 27 June 1995
LANAlert-AFG-Trap DEFINITIONS ::=
```

```
BEGIN

IMPORTS

enterprises      FROM RFC1155-SMI
OBJECT-TYPE      FROM RFC-1212
TRAP-TYPE        FROM RFC1215;
-- Network Computing Inc.
nci OBJECT IDENTIFIER ::= { enterprises 768 }
-- LANAlert alert packets
lanalert OBJECT IDENTIFIER ::= { nci 2 }
-- Agent-independent data items
lanalert-data OBJECT IDENTIFIER ::= { lanalert 2 }
-- (NOTE: Some MIB processors have problems with the definition
-- of lanalertFSA-NW2; this can be commented out if no
-- NetWare 2.x File Server Agents are in use.)
lanalert-agent OBJECT IDENTIFIER ::= { lanalert 3 }
lanalertFSA-NW2 OBJECT IDENTIFIER ::= { lanalert-agent 0 }
lanalertFSA-NW3o OBJECT IDENTIFIER ::= { lanalert-agent 1 }
lanalertNA OBJECT IDENTIFIER ::= { lanalert-agent 2 }
lanalertFSA-NW4o OBJECT IDENTIFIER ::= { lanalert-agent 3 }
lanalertAFG OBJECT IDENTIFIER ::= { lanalert-agent 4 }
lanalertFSA-NT OBJECT IDENTIFIER ::= { lanalert-agent 6 }
lanalertSNMPMon OBJECT IDENTIFIER ::= { lanalert-agent 7 }
lanalertMS OBJECT IDENTIFIER ::= { lanalert-agent 10 }
lanalertFSA-NW3 OBJECT IDENTIFIER ::= { lanalert-agent 50 }
lanalertFSA-NW4 OBJECT IDENTIFIER ::= { lanalert-agent 51 }
```

Agent-independent Data

LANAlert alerts are assigned one of five priorities, from 1 (highest) through 5 (lowest). The following values are used for the specific-trap field of AFG Trap protocol data units (PDU) to represent the various priorities on set-alert and clear-alert messages. Pre-2.4.0 Management Servers do not identify the alert priority when sending clears, so the value **clear-unknown** is used as the specific-trap number in this case. Otherwise, one of the values **clear-1** through **clear-5** is used to communicate the priority of a clear-alert message.

```
    LANAlertPriority ::= INTEGER {
                set-1(1),
                set-2(2),
                set-3(3),
                set-4(4),
                set-5(5),
                clear-unknown(6),
                clear-1(7),
                clear-2(8),
                clear-3(9),
                clear-4(10),
                clear-5(11)

    agentName OBJECT-TYPE
        SYNTAX DisplayString (SIZE (1..15))
        ACCESS not-accessible
        STATUS mandatory
        DESCRIPTION
          "The name of an agent reporting to a management server."
        ::= { lanalert-data 1 }

    nodeName OBJECT-TYPE
        SYNTAX DisplayString (SIZE (1..15))
        ACCESS not-accessible
        STATUS mandatory
        DESCRIPTION
          "The name of a node on the monitored network."
        := { lanalert-data 2 }

    eventID OBJECT-TYPE
        SYNTAX INTEGER (0..4294967295)
        ACCESS not-accessible
```

```
    STATUS mandatory
    DESCRIPTION
      "A number designating a monitored condition."
    := { lanalert-data 3 }

thresholdID OBJECT-TYPE
    SYNTAX INTEGER (1..4294967295)
    ACCESS not-accessible
    STATUS optional
    DESCRIPTION
      "A number designating a threshold set on a
       monitored condition."
    := { lanalert-data 4 }

alertText OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..79))
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
      "A string describing an alert condition."
    := { lanalert-data 5 }

managementServerName OBJECT-TYPE
    SYNTAX DisplayString (SIZE (1..15))
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
      "The name of a LANAlert management server."
    := { lanalert-data 6 }

nodeAddressIPX OBJECT-TYPE
    SYNTAX OCTET STRING (SIZE (12))
    ACCESS not-accessible
    STATUS optional
    DESCRIPTION
      "The IPX network address of a node."
    := { lanalert-data 7 }

nodeAddressAppleTalk OBJECT-TYPE
    SYNTAX OCTET STRING (SIZE (4))
    ACCESS not-accessible
    STATUS optional
    DESCRIPTION
      "The AppleTalk network address of a node."
    := { lanalert-data 8 }

nodeAddressIP OBJECT-TYPE
    SYNTAX OCTET STRING (SIZE (4))
    ACCESS not-accessible
    STATUS optional
    DESCRIPTION
      "The IP network address of a node."
    := { lanalert-data 9 }

alertType OBJECT-TYPE
    SYNTAX INTEGER {
            thresholdAlert(1),
            changeAlert(2),
            resettableAlert(3)
        }
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
      "The type of LANAlert alert packet.
```

Threshold alerts are generated when a condition crosses a preconfigured threshold, and are cleared by the agent when the condition crosses the preconfigured reset value.

Change alerts are generated when a condition changes state. These types of alerts are forwarded to any consoles and gateways that are currently attached to the agent management server. Change alerts cannot be cleared, since neither the agent or the management server maintains information about the alert (other than logging the alert). Console operators dismiss change alerts locally.

Resettable alerts are generated when a condition changes in a predefined manner. Resettable alerts can be cleared by a console operator, or by the agent itself for some alerts.

```
    lanalertFSA-NW3-s1 TRAP-TYPE
        ENTERPRISE lanalertFSA-NW3
        VARIABLES { managementServerName,
                nodeName,
                eventID,
                alertText
        DESCRIPTION
          "The LANAlert File Server Agent on NetWare 3.x has
          set a priority 1 alert."
        := 1  -- set-1
```

Class Definition Statement File Changes

The following is the entry for **lanalertFSA-NW3-s1** in the **tecad_snmp.cds** file:

```
CLASS lanalertFSA-NW3-s1
  SELECT
    1:ATTR(=,$ENTERPRISE),VALUE(PREFIX, "1.3.6.1.4.1.768.2");
    2:$SPECIFIC=1;
    3:ATTR(=,"managementServerName");
    4:ATTR(=,"nodeName");
    5:ATTR(=,"eventID");
    6:ATTR(=,"alertText");
  MAP
    managementServerName=$V3;
    nodeName=$V4;
    eventID=$V5;
    alertText=$V6;
    msg=PRINTF("The LANAlert File Server Agent on %s has set
a priority 1 alert.",$V4);
  END
```

The first line is the attribute or trap name. The first attribute (**1: ATTR(=,$ENTERPRISE) VALUE(PREFIX, ″1.3.6.1.4.1.768.2″)** ;) specifies that this is an enterprise trap. The OID prefix is derived from the trap definition; trap **lanalertFSA-NW3-s1** is of type **ENTERPRISE lanalertFSA-NW3**.

The enterprise OID prefix is 1.3.6.1.4.1 as specified in RFC1155-SMI, plus the appropriate object identifiers. From the following lines in the MIB file, the prefix can be expanded to 1.3.6.1.4.1.768.2:

```
nci             OBJECT IDENTIFIER ::= { enterprises 768 }
lanalert        OBJECT IDENTIFIER ::= { nci 2 }
```

The specific trap number is just a sequential numbering of trap definitions as defined in the MIB definition for **lanalertFSA-NW3-s1 TRAP-TYPE**. In this case **lanalertFSA-NW3-s1** is the first and is denoted as follows:

```
 2:$SPECIFIC=1;
```

The other attributes are derived from the trap expected object types. The definition for **lanalertFSA-NW3-s1** states that it contains the following information:

```
  VARIABLES { managementServerName,
      nodeName,
      eventID,
      alertText }
```

These are denoted in the **tecad_snmp.cds** file as follows:

```
  3:ATTR(=,"managementServerName");
  4:ATTR(=,"nodeName");
  5:ATTR(=,"eventID");
  6:ATTR(=,"alertText");
```

You would add the following entry to the **tecad_snmp.cds** file to map the trap variables to adapter variables:

```
  MAP
    managementServerName=$V3;
    nodeName=$V4;
    eventID=$V5;
    alertText=$V6;
    msg=PRINTF("The LANAlert File Server Agent on %s has set
a priority 1 alert.",$V4);
```

These variable values are then mapped to event attributes defined in the **tecad_snmp.baroc** file. For example, the BAROC class definition for the **lanalertFSA-NW3-s1** event is as follows:

```
TEC_CLASS :
    LANAlert_Trap ISA Specific_SNMP_Trap
    DEFINES {
        source:default="LANA";
        sub_source:default="NET";
        severity:default="WARNING";
        trapTime:INT32;
        specificTrap:INT32;
        managementServerName:STRING;
        nodeName:STRING;
        eventID:INT32;
        alertText:STRING;
        };
END

TEC_CLASS :
    lanalertFSA-NW3-s1 ISA LANAlert_Trap;
END
```

Object Identifier File Changes

The entry in the **tecad_snmp.oid** file for this trap is composed of the enterprise prefix plus the appropriate object identifiers (OID) plus the variable attribute OID. For example,

```
#nci                    1.3.6.1.4.1.768
lanalert                1.3.6.1.4.1.768.2
lanalert-data           1.3.6.1.4.1.768.2.2
nodeName                1.3.6.1.4.1.768.2.2.2
eventID                 1.3.6.1.4.1.768.2.2.3
alertText               1.3.6.1.4.1.768.2.2.5
managementServerName    1.3.6.1.4.1.768.2.2.6
```

Troubleshooting the SNMP Adapter

1. Use the following command to cold start the SNMP adapter:

   tecad_snmp [**–d**] [**–c** *configuration_file*]

   The following are the arguments for the **tecad_snmp** command:

–d Starts the adapter in debug mode. This argument prevents the daemon from forking itself.

–c *configuration_file*
Specifies the location of the configuration file.

If **–c** is not specified, then the adapter searches **$TECADHOME/etc/tecad_snmp.conf** if the environment variable **TECADHOME** is set, or **/etc/Tivoli/tecad/etc/tecad_snmp.conf** for the configuration file.

2. Make sure that there are no other processes such as SNMP or **ovtrapd** already listening on port 162. Use **netstat –a | grep 162** to see if this port is in use. The first process to start up gets the port and the other processes that follow never receive events from that port.
3. Use **snmptrap** or the Tivoli Distributed Monitoring **wsnmptrap** commands to send events to the adapter for testing.
4. Change all **/dev/null** entries in the **.err** file to the file name you want. Stop and restart the adapter, send an event through, and then look in the trace file to see what processing was done on the event.

Chapter 8. IBM Tivoli Enterprise Console Gateways

Although not an adapter, the IBM Tivoli Enterprise Console gateway is similar in that it is software that uses the TME interface of Tivoli Event Integration Facility to communicate with the event server. Like an adapter, it can be configured with a configuration file, and the configuration file can be distributed with an adapter configuration profile (ACP) entry using the Adapter Configuration Facility (ACF).

The IBM Tivoli Enterprise Console gateway and all of the necessary adapter files for each endpoint operating system are installed on the managed node when the ACF is installed on the managed node. The ACF is required to be installed on the same managed node as the Tivoli Management Framework gateway so adapters and adapter-related files can be distributed to endpoints. Therefore, it is important to install the ACF on every managed node that is configured as a Tivoli Management Framework gateway throughout a Tivoli management region.

Note: To distribute a modified IBM Tivoli Enterprise Console gateway ACP to a managed node with an IBM Tivoli Enterprise Console gateway installed, the managed node must also have an endpoint installed on it. When you distribute the profile, the subscriber must be the endpoint on that managed node.

See "How Events Get Sent to the Event Server" on page 1 for an overview of the IBM Tivoli Enterprise Console gateway, referred to in the rest of this chapter as the gateway.

Controlling Event Traffic at the Gateway

At certain times, the number of events coming from endpoint adapters can overwhelm the gateway, the event server, and even the network. With the gateway configuration file, you can control the number of events sent across the network. Thus, you can control the amount of event traffic in your environment.

You control the number of events sent from the gateway to the event server with the **EventSendThreshold**, **BufferFlushRate**, and **MaxGWCacheSizeMegs** keywords. To control the number of events sent from the adapter, see "Event Filtering" on page 14.

Example

To improve the event server performance, the following steps exemplify how to determine the values for the **BufferFlushRate** and **EventSendThreshold** keywords for the gateway.

The values provided in this example can vary greatly from installation to installation, depending on how many events, adapters, and gateways are in a particular environment. Use the worksheets provided in "Worksheets and Calculations" on page 97 to collect and calculate the data for your environment. All numerical values are expressed in events per second, except where noted.

1. Determine the average number of events that the event server can process.

   The example event server, on average, processes approximately 120 events per second without degrading its performance.

2. Determine the number of gateways and the resulting number of events that they can send to the event server.

   The example environment contains two gateways, where gateway A is responsible for Web commerce servers and gateway B is responsible for the secretaries' systems. Divide the average capacity of the event server by the number of gateways:

   ```
   120 ÷ 2 = 60
   ```

   The resulting value of 60 indicates the average number of events each gateway can send without overwhelming the event server. Continue with step 3 to obtain the adjusted values for the gateway send rate.

3. Calculate the value for the **EventSendThreshold** keyword.

   The **EventSendThreshold** keyword sets the maximum number of events per second that the gateway sends to the event server. Because gateway A forwards events from mission-critical systems, more gateway A events should be sent to the event server than gateway B events. Thus, the **EventSendThreshold** keyword for gateway A is set to 80 events per second. Gateway B has the **EventSendThreshold** keyword set to 40 events per second. In this way, more gateway A events get to the event server.

   The sum of the values for gateway A and gateway B must be less than or equal to the 120 events that the event server can process:

   ```
   80 + 40 ≤ 120
   ```

4. Determine the value for the **BufferFlushRate** keyword.

   Any events above the value specified for the **EventSendThreshold** keyword are stored in the cache on the gateway. To regulate the number of events being sent to the event server, the **BufferFlushRate** keyword controls the number of events per minute to be sent from the cache, when the gateway recovers a lost connection to the event server.

   For gateway A, the **BufferFlushRate** keyword is set to 5400 events per minute (90 events per second), and for gateway B the keyword is set to 3000 events per minute (50 events per second). Thus at peak traffic times, the event server is receiving 140 events per second from both gateways:

   ```
   90 + 50 = 140
   ```

   Although 140 events per second is greater than the average capacity of the event server (120 events per second), the event server has the capability to process excess events during brief, intermittent periods of time.

   Tip: Remember to convert events per second to events per minute before setting the value for the **BufferFlushRate** keyword.

5. Modify the gateway ACP with the values calculated in step 3 and step 4.
6. Distribute the gateway ACP.

Depending on the number of gateways and endpoints in your environment, you need to carefully consider the rates you specify for the keywords. For instance, an improper configuration might have multiple gateways sending events at the same rate, thus flooding the event server at the same time. See "Configuration File" on page 97 for details about these keywords.

Worksheets and Calculations

Table 1 and Table 2 summarize the values for this example. You can use these tables as worksheets to assemble the values you measure and calculate for your environment. All numerical values are expressed in events per second, except where noted.

Table 1. Example values for controlling event traffic for the event server

Average Receive Rate	Expected Peak Rate for High Traffic
120	140

Table 2. Example values for controlling event traffic for gateways

	Event Send Rate	Adjusted Rate	EventSendThreshold	BufferFlushRate
Gateway A	60	80	80	5400 events per minute (90 events per second)
Gateway B	60	40	40	3000 events per minute (50 events per second)
Total Events Sent to Event Server	120	120	120	140 events per second

The following are the calculations to control event traffic:

```
event server average rate ≥ gateway A events + gateway B events

EventSendThreshold = adjusted send rate for gateway

   gateway A         gateway B
BufferFlushRate + BufferFlushRate ≤ event server peak rate
```

Additionally, you can control event traffic with state correlation provided with Tivoli Event Integration Facility and the tec_gateway_sce ACP. See the *Tivoli Event Integration Facility User's Guide* for more information about filtering events with state correlation.

Configuration File

The gateway configuration file is optional and does not exist on the managed node until an ACP containing gateway configuration information is distributed to the endpoint on the managed node. Default values are in effect until they are modified by distributing an ACP containing gateway configuration information.

The configuration file names and their locations are as follows:

UNIX:

```
/etc/Tivoli/tec/tec_gateway.conf
```

Microsoft Windows:

```
%SystemRoot%:\drivers\etc\Tivoli\tec\tec_gateway.conf
```

The following example illustrates how the Windows path notation can be expanded:

```
c:\winnt\system32\drivers\etc\Tivoli\tec\tec_gateway.conf
```

The configuration file defines the behavior of the gateway. The configuration file can have the common keywords described in “Keywords” on page 9, as well as the following custom keywords:

BufEvtPath Specifies the gateway to buffer events at this location if it cannot forward them to the event server. Because a single gateway can forward events to multiple event servers, it must have an event buffer file for each of those event servers. This allows the gateway to send events to the correct event servers when it re-establishes connections to them and flushes buffers. These buffer files are created by the gateway appending the event server location to the name of the file defined with the **BufEvtPath** option.

The following example shows how to specify this option:

```
ServerLocation=@EventServer#tmr-central
BufEvtPath=/etc/Tivoli/tec/gateway_cache
```

In the example, the actual file created is **/etc/Tivoli/ tec/gateway_cache@EventServer#tmr-central**.

More than one buffer file might be created at the gateway, depending on how many event server locations are configured by the adapters sending events. For each different server location, a separate buffer file is created. Continuing with the example, if the gateway received an event from an adapter that specified server location **@EventServer#tmr-east**, an additional file named **/etc/Tivoli/tec /gateway_cache@EventServer#tmr-east** would be created.

The default is the following:

UNIX: /etc/Tivoli/tec/cache@EventServer#*region*

Windows: $DBDIR/cache.dat@EventServer#*region* (on managed nodes)

$TIVOLIHOME/tec/$(AC_TYPE).cache@ EventServer#*region* (on endpoints)

Note: The keywords **Filter**, **FilterCache**, and **FilterMode** can be used for the gateway. For reducing network load and improving performance of the managed node containing the gateway, it is better to filter events as close to the source as possible; that is, by specifying filter options in adapter configuration files.

EventSendThreshold
Specifies the maximum number of events per second to send to the server. Use this parameter with the **BufferFlushRate** keyword.

GatewayAckInterval
Specifies the timeout interval, in seconds, to wait for the

acknowledgement from the event server. The default value is 30 seconds. This keyword works with the **GatewayTMEAckEnabled** keyword for event delivery.

GatewayQueueSize

Specifies, in bytes, the size for the buffers containing events waiting to be forwarded to event servers. If any of these buffers fill before the expiration of the **GatewaySendInterval** option, the waiting events are immediately sent.

The default is 40 000 bytes.

GatewaySendInterval

Specifies, in seconds, the interval to forward events to event servers. The gateway holds events when they are received, and then bundles them up and forwards them in a message to the appropriate event server when this interval expires.

The default is five seconds.

GatewayTMEAckEnabled

When set to YES, connection-oriented, TME connections expect an acknowledgement from the event server before the gateway discards the sent events. Use this keyword to ensure the event delivery. The default value is NO.

MaxGWCacheSizeMegs

Specifies the maximum size in megabytes that the cache file can grow to. By default, the maximum size of the cache is 1 MB. The keyword does not appear in the configuration file.

RetryInterval When events cannot be sent to an event server, the gateway waits the specified number of seconds before connecting to a secondary server. While the gateway is waiting for the expiration of this interval, new events continue to be received by the gateway and are buffered in memory (and cached to disk as necessary). You can adjust the size of the gateway memory queues for adapters with the **GatewayQueueSize** keyword.

The **RetryInterval** option allows adapters to send all events to the primary event server even if the primary event server is stopped briefly, such as when loading a new rule base. If you use this option to wait for the restarting of an event server, set the value for a period of time longer than needed for the event server to be stopped and restarted.

The **RetryInterval** keyword is optional.

ServerLocation

Specifies a default event server location to be used if adapters do not specify a server location in an event instance. If **ServerLocation** is specified in the configuration file for an adapter, that location is passed from the adapter to the gateway as part of the event, and the gateway forwards the event to the adapter-specified event server. If **ServerLocation** is not specified in an adapter configuration file, the gateway sends the events for that adapter to the event server specified with the **ServerLocation** keyword. If **ServerLocation** is not specified in the gateway configuration file, the gateway sends the event to the event server in the local Tivoli management region.

The default value is @EventServer.

You can specify multiple server names as a comma-delimited list. Server names later in the list can be backups for times when the gateway cannot contact its primary server for an event and the **RetryInterval** has expired without successfully contacting the primary server. You can specify a host name as you would for a non-TME adapter, and the events are then forwarded to that host using the non-TME Tivoli Event Integration Facility. For more details, see the examples in the **ServerPort** description.

When the gateway cannot contact the adapter-specified event server, the server names specified in the list are backups. If **ServerLocation** is not present in the gateway configuration file, the backup functionality uses the default @EventServer value.

To prevent event delivery to backup servers, set the **ServerLocation** keyword to NONE.

ServerPort Specifies the port for the event server when sending events using the non-TME Tivoli Event Integration Facility. This keyword is ignored if you are not using the non-TME Tivoli Event Integration Facility.

The default value is zero (0). A value of zero tells the non-TME Tivoli Event Integration Facility to contact the portmapper on the specified host to determine where the event server is listening for incoming event. Note that if you are forwarding events to a Tivoli Availability Intermediate Manager, you cannot specify zero (0) as the port because the Tivoli Availability Intermediate Manager does not register itself with the portmapper.

If you specify a value for **ServerPort**, the value must be either one integer value or a comma-delimited list containing the same number of values as the list of event servers specified in **ServerLocation**. You can use one integer value to apply to all of the event servers listed in **ServerLocation**; otherwise, each event server in **ServerLocation** requires a corresponding value in **ServerPort**.

The following example shows how to specify multiple server names to use backup servers in case the primary server fails to receive events. Suppose you have a Tivoli Availability Intermediate Manager running on hosts aim.xyz.com and aimbkup.xyz.com, and **ServerLocation** and **ServerPort** are specified as follows:

```
ServerLocation=aim.xyz.com,aimbkup.xyz.com,@EventServer
ServerPort=5530,5531,0
```

This sends events to port 5530 on the host aim.xyz.com using the non-TME Tivoli Event Integration Facility. If that fails, events are sent to port 5531 on the host aimbkup.xyz.com. If that also fails, events are sent directly to the event server using the TME Tivoli Event Integration Facility. The port value of zero (0) specified for @EventServer is ignored because port numbers are not needed with the TME Tivoli Event Integration Facility.

Chapter 9. UNIX Log File Adapter

The TME UNIX log file adapter receives raw log file information from the UNIX **syslogd** daemon, formats it, and sends it to the IBM Tivoli Enterprise Console gateway. The IBM Tivoli Enterprise Console gateway then sends the information to the event server. The non-TME UNIX log file adapter sends information directly to the event server.

The UNIX log file adapter adds entries into the **/etc/syslog.conf** file to enable the adapter to monitor events that the **syslogd** daemon writes to various log files. The adapter can also be configured to monitor any ASCII log file for information that is important to the operation of your enterprise.

The UNIX log file adapter can only parse log files that create raw event information in single-line form for each event. You must preprocess log files that contain raw event information in multiple-line form or if the update quantity or rate is extremely high.

This chapter explains how to configure and start the UNIX log file adapter.

Event Server Configuration

At the event server, the BAROC file and rule set file must be imported into a rule base and then compiled. This rule base must then be loaded and made the active rule base. See the *IBM Tivoli Enterprise Console Rule Builder's Guide* for additional information about the steps to do these tasks.

Note: The **Default** rule base, as shipped, is already configured using the BAROC file and default rule file for the UNIX log file adapter.

Starting the Adapter

Use the **init.tecad_logfile start** command in the background to manually start the adapter. Always use this command to ensure that the **syslogd** daemon is properly configured to send messages to the adapter.

In most situations, the start-up process takes 40 seconds, at which time the **syslogd** daemon is refreshed. If you want to give the adapter additional seconds to complete its startup, specify the **–t***startup_time* option for the **init.tecad_logfile start** command. There cannot be a space between the option letter and the option value. This option is useful if the adapter does not receive events because the **syslogd** daemon is not properly refreshed.

Note: The endpoint adapter is automatically started as a step in the adapter installation process when the adapter configuration profile (ACP) is distributed using the Adapter Configuration Facility (ACF).

Stopping the Adapter

Use the **init.tecad_logfile stop** command to manually stop the adapter. Always use this command to ensure that the **syslogd** daemon is correctly configured to stop sending messages to the adapter. If the adapter is stopped with any other method, the **syslogd** daemon might exit because the adapter is no longer listening on the named pipe the **syslogd** daemon is writing to.

Note: The endpoint adapter can be automatically stopped by distributing an ACP that has the adapter start command removed from the after-file-distribution actions. See the *IBM Tivoli Enterprise Console User's Guide* for additional information.

Running Multiple UNIX Log File Adapters

You can run multiple instances of the UNIX log file adapter on a single system. It is recommended that additional adapters be run as non-TME adapters. To monitor different log files, each instance of the adapter must have its own configuration, format, class definition statement (CDS), and error files.

If you want to stop an adapter when multiple log files are running, you must specify the name of the adapter to stop. If you do not specify the adapter to stop, the default adapter without a name is stopped.

The syntax for the **init.tecad_logfile** command is the following:

init.tecad_logfile [**–s**] {**start** | **stop**} [*AdapterID*] **&**

If the **–s** flag (skip syslog) is specified, the adapter does not monitor the **syslogd** daemon.

If the **–s** flag is not specified, use **&** so that the command runs in the background while returning a command prompt to your session. Otherwise, because an adapter started without the **–s** option forks a child process to run the adapter, the process does not return to the command line until the child process ends.

Note: If you start the adapter with the **–s** flag, you can also use the **–s** flag when you stop the adapter to avoid reconfiguring the **syslogd** daemon. You can also stop the adapter without the **–s** flag and it still works. However, do not stop an adapter with the **–s** flag if you did not start it with the **–s** flag.

If the **–s** flag is not specified, the UNIX log file adapter startup script uses a UNIX pipe to monitor the **syslogd** daemon and the **syslogd** daemon is configured to write to the pipe, and the UNIX log file adapter reads from that pipe. When the adapter ends, the startup script will reconfigure the **syslogd** daemon to stop writing to the pipe before stopping the UNIX log file adapter.

Do not configure more than one UNIX log file adapter to read from the **syslogd** daemon on a particular system.

The following command starts a UNIX log file adapter called **syslog** that monitors all syslog messages:

```
init.tecad_logfile start syslog &
```

Adapter Files

The UNIX log file adapter package consists of the following files:

tecad_logfile.cfg
The installation script.

init.tecad_logfile
The adapter startup and shutdown script. Never stop the adapter using signals. Use this script to ensure that the **syslogd** daemon remains running and functional.

tecad_logfile The executable file of the adapter that receives the log information and transforms it into events.

logfile_gencds
The executable file that converts a format file to a CDS file.

tecad_logfile.baroc
The BAROC file.

tecad_logfile.cds
The CDS file. This file is created by running **logfile_gencds** on the format file.

tecad_logfile.conf
The configuration file.

tecad_logfile.err
The error file.

tecad_logfile.fmt
The format file.

log_default.rls
The default rule file.

Before you start the event server and UNIX log file adapter, check each adapter file to determine if it defines the behavior you want from the adapter.

Configuration File

The configuration file defines the behavior of the adapter. The configuration file can have the common keywords described in “Configuration File” on page 9, as well as the following custom keywords:

LogSources
Specifies the log files to poll. The complete path to each file must be specified, and file names must be separated by commas; no spaces or other separators can be used. A log source need not exist when the adapter is started; it will be polled when it is created.

If a file truncates while the adapter is active, the adapter automatically resets its internal pointer to the beginning of the file. If during the polling interval the file is overwritten, removed, or recreated with more lines than the previous poll, only the number of lines greater than the previous line count is read. For example, the file has one line. After the poll interval elapses, the file is overwritten with two lines. Only the second line is read on the next polling.

Note: The maximum number of lines that can be concatenated to a log file is 16 384.

PollInterval
Specifies the frequency, in seconds, to poll each file listed in the **LogSources** field for new messages. The default value is 120 seconds.

UnmatchLog
Specifies a file to log discarded events that cannot be parsed into an IBM Tivoli Enterprise Console event class by the adapter. The discarded events can then be analyzed to determine if modifications are needed to the adapter format file.

Format File

The format file is described in detail in "Format File" on page 17.

Class Definition Statement File

The CDS file defines how an adapter constructs events. This file is derived from the format file using the **logfile_gencds** program. In general, you should never have to edit this file to add new mappings. The CDS file is described in detail in "Class Definition Statement File" on page 18 and in Appendix C, "Class Definition Statement File Reference" on page 155.

Error File

The error file is described in detail in "Error File" on page 19.

Events Listing

The following table shows the class names and severities of all events defined for the UNIX log file adapter. You can use the table to get a sense of how log file events are mapped to IBM Tivoli Enterprise Console events and to determine if you want to make any changes. The events are defined in the BAROC file. See the *IBM Tivoli Enterprise Console Rule Builder's Guide* for more information about customizing **.baroc** files.

Event Class Structure

Event classes are defined hierarchically, with child classes inheriting attribute value defaults from the parent.

The adapter fills in the following attribute defaults. The attributes are used in event group filters.

- **source**: LOGFILE
- **origin**: *hostIPaddress*
- **hostname**: *hostname*

The following events are defined for the UNIX log file adapter in the **tecad_logfile.baroc** file.

Event Class	Default Severity
Logfile_Base	WARNING
Logfile_Automounter	HARMLESS

Event Class	Default Severity
Logfile_Amd	WARNING
Amd_Mounted	WARNING
Amd_Unmounted	WARNING
Logfile_Automount	WARNING
Logfile_Bootpd	WARNING
Logfile_Comsat	WARNING
Logfile_Cron	HARMLESS
Logfile_Date	HARMLESS
Logfile_Date_Set	WARNING
Logfile_Ebbackupd	WARNING
Ebbackupd_Waiting	WARNING
Logfile_Ebcatcomp	WARNING
Logfile_Fsck	WARNING
Logfile_Ftp	WARNING
Logfile_Ftpd	WARNING
Logfile_Gated	WARNING
Logfile_Getty	WARNING
Logfile_Halt	WARNING
Logfile_Idi	HARMLESS
Logfile_Inetd	WARNING
Logfile_Init	WARNING
Logfile_Innd	WARNING
Logfile_Kernel	WARNING
File_Write_Error	MINOR
File_System_Full	MINOR
NFS_Write_Error	WARNING
Sendsig_Err	CRITICAL
Kernel_Panic	FATAL
NFS_No_Response	WARNING
NFS_OK	HARMLESS
Silo_Overflow	MINOR
Logfile_Login	WARNING
Root_Login	MINOR
Root_Login_Failure	WARNING
Root_Login_Failure_From	WARNING
Root_Login_Success	WARNING
Root_Login_Success_From	WARNING
Repeated_Login_Failure	WARNING
Repeated_Login_Failure_From	WARNING
Logfile_Lpd	WARNING

Event Class	Default Severity
Logfile_Lpd_Get_Hostname	WARNING
Logfile_Lpd_Lost_Connection	WARNING
Logfile_Lpd_No_File	WARNING
Logfile_Mosaic	WARNING
Logfile_Mountd	WARNING
Logfile_Named	WARNING
Logfile_Nfsd	WARNING
Logfile_Nnrpd	WARNING
Logfile_Oserv	WARNING
Oserv_Panic	CRITICAL
Oserv_Graceful_Exit	HARMLESS
Oserv_System_Error	MINOR
Oserv_Fork_Failed	CRITICAL
Oserv_Exec_Failed	MINOR
Oserv_Comm_Error	WARNING
Oserv_IPC_Dispatch_Failed	MINOR
Oserv_Security	WARNING
Oserv_Tmgr	WARNING
Oserv_Event_Method_Failed	MINOR
Logfile_Passwd	WARNING
Logfile_Pcnfsd	WARNING
Logfile_Printer	WARNING
Printer_Connection_Abort	WARNING
Printer_Error_Cleared	HARMLESS
Printer_Door_Open	WARNING
Printer_Offline	WARNING
Printer_Output_Full	WARNING
Printer_Page_Punt	WARNING
Printer_Paper_Jam	WARNING
Printer_Paper_Out	WARNING
Printer_Powerup	WARNING
Printer_Toner_Low	WARNING
Logfile_Rarpd	WARNING
Logfile_Reboot	HARMLESS
Logfile_Rexecd	WARNING
Logfile_Rftp	WARNING
Logfile_Rlogind	WARNING
Logfile_Routed	WARNING
Logfile_Rquotad	WARNING
Logfile_Rshd	WARNING
Logfile_Rstatd	WARNING

Event Class	Default Severity
Logfile_Rtelnet	WARNING
Logfile_Rwhod	WARNING
Logfile_Sendmail	HARMLESS
Sendmail_Loopback	WARNING
Sendmail_No_Space	MINOR
Logfile_Snmpd	WARNING
Logfile_Sockd	WARNING
Sockd_Connected	HARMLESS
Sockd_Terminated	WARNING
Sockd_Transfer	WARNING
Logfile_Strerr	HARMLESS
Logfile_Su	WARNING
Su_Failure	WARNING
Su_Success	WARNING
Logfile_Syslogd	WARNING
Syslogd_Nospace	MINOR
Logfile_Talkd	WARNING
Logfile_Telnetd	WARNING
Logfile_Tftpd	WARNING
Logfile_Xntpd	WARNING
Xntpd_Clock_Reset	WARNING
Xntpd_Ntpdate	WARNING
Logfile_YP	HARMLESS
Logfile_Ypbind	WARNING
Logfile_Ypchfn	WARNING
Logfile_Ypchsh	WARNING
Logfile_Yppasswd	WARNING
NIS_No_Response	WARNING
NIS_OK	HARMLESS
No_Permission	WARNING
No_Resources	CRITICAL
No_Disk_Space	WARNING
File_System_Full	MINOR
LOCAL_File_System_Full	WARNING
NFS_File_System_Full	WARNING
SWAP_File_System_Full	WARNING
Sendmail_No_Space	MINOR
Syslogd_Nospace	MINOR
No_Memory	WARNING
No_Proc_Attributes	WARNING
Server_No_Response	WARNING

Event Class		Default Severity
	NFS_No_Response	WARNING
	NIS_No_Response	WARNING
Server_OK		HARMLESS
	NFS_OK	HARMLESS
	NIS_OK	HARMLESS

Default Rules

The UNIX log file adapter has a set of default rules that can be installed to enhance event server operation. Rules can enable the server to perform functions such as deleting events and sending e-mail to alert administrators of an unresolved problem. The rules are contained in the **log_default.rls** file and perform the following functions:

- Duplicate events of the following classes are filtered out and the first event repeat count is incremented:
  - **Printer_Paper_Out**
  - **Printer_Toner_Low**
  - **Printer_Offline**
  - **Printer_Output_Full**
  - **Printer_Paper_Jam**
  - **Printer_Door_Open**
- Printer assistance can be called for when a printer condition persists for a period of time greater than 90 seconds. If any of the following conditions persist for that period of time, an e-mail message is sent to the e-mail alias **tec_print** in order to request assistance with the printer condition. (The **tec_print** alias must be added to the e-mail alias file before the messages can be delivered.)
  - **Printer_Paper_Out**
  - **Printer_Toner_Low**
  - **Printer_Offline**
  - **Printer_Output_Full**
  - **Printer_Paper_Jam**
  - **Printer_Door_Open**
- When a printer condition is cleared, the event server automatically closes the event that indicated a problem. If e-mail was sent out notifying the administrators of the printer problem, the server sends e-mail indicating the condition has cleared up.
- The **Su_Success** and **Su_Failure** events indicate that a user attempted to use the **su** command. If a **Su_Success** event is received within 90 seconds of the **Su_Failure** event, the server assumes that the **Su_Failure** was a mistake and downgrades the event to **HARMLESS** and closes the **Su_Failure** event. The rules ensure that these two events are related by checking that they occurred on the same host, the user attempting this was the same, and the user that they were trying to change to was the same.
- Some of the log file events are relevant for a short amount of time. The administrators also do not want to be burdened with closing these events manually. A rule is provided that closes the following event classes after one

hour. You can edit this rule to change the time or the list of classes. Refer to the *IBM Tivoli Enterprise Console Rule Builder's Guide* for information about editing rules.

  - **Logfile_Amd**
  - **Logfile_Cron**
  - **Logfile_Oserv**
  - **Logfile_Date_Set**

The event server also comes with some additional rules that you can install. The **$BINDIR/TME/TEC/contrib/rules/security** directory contains the **security_default.rls** file, which provides the following behavior to the event server:

- When a host reports a repeated login failure attempt at least two times in a row, e-mail is sent to the e-mail alias **tec_security** notifying the administrators of the attempted security breach. (The **tec_security** alias must be added to the e-mail alias file before the messages can be delivered.)
- A rule is included that closes the following event classes after one hour:
  - **Repeated_Login_Failure**
  - **Repeated_Login_Failure_From**
  - **Root_Login_Success_From**

Troubleshooting the UNIX Log File Adapter

Perform the following steps to troubleshoot the UNIX log file adapter:

1. Stop any UNIX log file adapters that are currently running:

   ```
   init.tecad_logfile stop
   ```

2. Start the adapter in debug mode.

   ```
   init.tecad_logfile -d start
   ```

3. Generate some messages to determine if the adapter receives them. You can send e-mail, perform an **su**, or perform any action that results in a write to syslog. Alternatively, you can use the **logger** program to generate messages:

   ```
   logger -t oserv -i execve failed: path: errno 13
   ```

   This generates an **Oserv_Exec_Failed** event. The message written by logger should match one of the format specifications in the **tecad_logfile.fmt** file.

4. When events arrive, the adapter prints messages to the screen indicating the class and the attribute values in the class.

   ```
   matched CREATED_PROFILE_MANAGER name is 'Profile1''
   ```

   If you do not see any messages, the adapter is not receiving events from the log file.

   Verify that the **syslogd** daemon is running and is writing any new messages to the system log files in **/var/adm** or its equivalent, or to the system console, depending on how **syslog.conf** has been configured to write out messages. For testing purposes, you can temporarily add the following line to **syslog.conf**:

   ```
   *.info <Tab> <filename>
   ```

   This allows all messages to be written to a file so you can see what messages have arrived. This file grows large quickly, so make this a temporary change only. You need to HUP the **syslogd** daemon each time you change **syslog.conf** to put these changes into effect.

5. If you see the messages, the adapter is receiving events and processing them. Run the **wtdumprl** command on the event server and verify that the messages are actually showing up in the reception log. If not, the events were not received by event server or there is a problem with the event server reception process. Check the adapter configuration file to verify that **ServerLocation** and **ServerPort** are properly defined. If the event class appears in any filter entry in the configuration file, it is not sent to event server. The administrator who started the adapter must have the required roles if you are running the TME version of the adapter. For a TME adapter, running the **odstat** command can offer some clues as to what failed.
6. If the reception log has a **PARSING_FAILED** error, the BAROC definition of the class does not match the event that is being received from the adapter. Usually the error messages pinpoint the problem.
7. If the previous steps do not indicate any problem and you do not see the new events in the IBM Tivoli Enterprise Console product, there might be a problem with the event group filters. Make sure the class filters match the classes in the BAROC files.
8. Change all **/dev/null** entries in the **.err** file to the file name you want. Stop and restart the adapter, send an event through, and then look in the trace file to see what processing was done on the event.

Chapter 10. Windows Event Log Adapter

The adapter for the Microsoft Windows event log forwards events from a Windows system to the event server. It is registered with the start-up configuration of Windows 2000 or Windows NT so that the adapter is started with all the other applications that are automatically started when Windows is started.

The adapter is a WIN32 process that reads events generated on a Windows 2000 or Windows NT system, formats them according to the specification in the **format** file, and forwards them using Winsock TCP/IP to an event server for further processing.

Events are gathered from up to six Windows event logs (System, Application, Security, DNS server, File Replication service, and Directory service) maintained by the Windows Event Manager, and from any other ASCII log files residing on the Windows 2000 or Windows NT system. The Windows event log adapter tracks the messages read from the Windows event logs using up to six registry variables that contain the most recent highest message read for the System, Application, Security, DNS server, File Replication service, and Directory service logs, whether the Windows event log adapter is running continuously or is restarted. You can alter this behavior using the appropriate switches when the Windows event log adapter is started.

Two versions of the Windows event log adapter are provided. One is built as a Windows service, while the other is a WIN32 process that is a command line interface version. Normally, you should run the Windows service version, since it runs even when no user is logged in. The command line interface can be used to help you view console messages for diagnostic purposes. Other than the service-related differences, both versions perform identically.

This chapter describes how to configure and start the Windows event log adapter.

Adapter Files

The Windows event log adapter package consists of the following files:

README The readme file.

tecinstl_win.cmd
The adapter installation batch file.

instlsrv.exe The adapter installation assist executable file.

tecadwins.exe
The adapter service executable file.

tecad_win.exe
The adapter non-service executable file.

tecad_win.conf
The configuration file.

tecad_win.fmt
The format file.

tecad_win.cds The class definition statement (CDS) file.

tecad_win.baroc
The BAROC file.

postemsg.exe The command line interface program to send an event to an event server.

tecad_win.err The error file.

Before starting the event server, check the configuration file to determine if it defines the preferred adapter behavior.

Configuration File

The configuration file defines the behavior of the adapter. This file can contain the common keywords described in "Configuration File" on page 9, as well as the following adapter-specific keywords:

HostnameIsAdapterHost
Specifies whether the hostname attribute for Windows NT Event Log events is set to the host on which the adapter is running (the default) or the host where the event originated.

If set to **NO** or **no**, the hostname attribute is set to the **COMPUTER** field from the Windows NT Event Log.

Note: This only applies to events from the Windows NT Event Log, not those generated from log files specified in **LogSources**. Those events always have the hostname attribute set to the host on which the adapter is running.

The **COMPUTER** name returned from the Windows NT Event Log might not be the same as the **ManagedNode** name (which is case-sensitive) of the host where the event originated. You must take this into consideration if you run tasks or programs from the IBM Tivoli Enterprise Console product or the rule base, because they might use the hostname attribute to determine where they run.

LanguageID Sets the language event log messages to be formatted in English or the native language. Valid values are one of the following:

ENGLISH
Messages are formatted in English.

DEFAULT
The adapter attempts to format event log messages in the default language based on the local value set in Windows. If the adapter cannot use the default language, it uses English. The value **DEFAULT** can only be used in languages that have 8-bit wide characters.

The format file is in English. The Windows event logs are in your native language. If your native language is not English, you must rewrite the format file in your native language.

LogSources Specifies the ASCII log files to poll for messages. The complete path to each file must be specified, and file names must be separated by commas; no spaces or other separators can be used. A log file source need not exist when the adapter is started; it will be polled when it is created.

If a file truncates while the adapter is active, the adapter automatically resets its internal pointer to the beginning of the file. If during the polling interval the file is overwritten, removed, or recreated with more lines than the previous poll, only the number of lines greater than the previous line count is read. For example, the file has one line. After the poll interval elapses, the file is overwritten with two lines. Only the second line is read on the next polling.

NumEventsToCatchUp

Specifies which event in the Windows event logs that the adapter starts with. This option provides some flexibility if the source being monitored is new or the adapter has been stopped for an extended period of time. Valid values are as follows:

0 Start with the next event in the logs.

–1 Start with the oldest event in the logs.

n *n* represents any number other than zero (0) or –1. Start with the *n*th event from the most current event in the logs; that is, start *n* events back from the most current event in the logs. If *n* is greater than the number of events that are available, all the events that are available are processed.

PollInterval Specifies the frequency, in seconds, to poll each log file listed in the **LogSources** keyword for new messages. The default value is 120 seconds.

PreFilter Specifies how events in a Windows event log are filtered before adapter processing. **PreFilter** statements are used by **PreFilterMode** when determining which events are sent from an event log to the adapter. An event matches a **PreFilter** statement when each *attribute=value* specification in the **PreFilter** statement matches an event in the event log. A **PreFilter** statement must contain at least the log specification and can contain up to three additional specifications, which are all optional: event ID, event type, and event source. The order of the attributes in the statement does not matter.

The basic format of the **PreFilter** statement is as follows:

```
PreFilter:Log=log_name;EventId=value; EventType=value;Source=value;
```

You can specify multiple values for each attribute by separating each with a comma.

Each **PreFilter** statement must be on a single line.

You can also use Tcl regular expressions in a **PreFilter** statement. The format of a regular expression is `re:'value_fragment'`.

Note: The IBM Tivoli Enterprise Console product uses one exception to the Tcl regular expression syntax. The backslash character (\) in the IBM Tivoli Enterprise Console product means the literal character that follows is the character to filter for, not some special character such as a tab. For example, `\t` means the tab character in Tcl but means `t` in the IBM Tivoli Enterprise Console product.

The following example shows a **PreFilter** statement with a regular expression. This prefilter statement matches all Application Log events with a source name that contains TEC_ somewhere in its name:

```
PreFilter:Log=Application;Source=re:'TEC_.*';
```

The following example shows a prefilter statement with a more narrow range. This prefilter statement matches all Application Log events with a source name that contains TEC_ somewhere in its name *and* has an **EventID** of 24:

```
PreFilter:Log=Application;Source=re:'TEC_.*';EventID=24;
```

For more information about Tcl regular expressions, see a Tcl user's guide.

The **PreFilter** keyword is optional. All Windows log events are sent to the adapter if prefilters are not specified. and **PreFilterMode=OUT**.

For additional information about prefiltering Windows log events, see "Prefiltering Windows Log Events" on page 115.

PreFilterMode

Specifies whether Windows log events that match a **PreFilter** statement are sent (**PreFilterMode=IN**) or ignored (**PreFilterMode=OUT**). Valid values are **IN**, **in**, **OUT**, or **out**. The default is **OUT**.

The **PreFilterMode** keyword is optional; if **PreFilterMode** is not specified, only events that do not match any **PreFilter** statements are sent to the adapter.

Note: If you set **PreFilterMode=IN**, make sure you have one or more **PreFilter** statements defined as well.

For additional information about prefiltering Windows event log events, see "Prefiltering Windows Log Events" on page 115.

SpaceReplacement

When **SpaceReplacement** is **FALSE**, any spaces in the security ID and subsource fields of the event log messages are left unchanged. When **SpaceReplacement** is **TRUE**, any spaces in the security ID and subsource fields of the event log messages are replaced with underscores. Set **SpaceReplacement** to **TRUE** if the format file expects the security ID and subsource fields to be a single word (that is, uses a **%s** format specification for them). The default setting is **FALSE**.

UnmatchLog Specifies a file to log discarded events that cannot be parsed into a IBM Tivoli Enterprise Console event class by the adapter. The discarded events can then be analyzed to determine if modifications are needed to the adapter format file.

WINEVENTLOGS

Controls which Windows Event Logs are monitored; also controls the service version and overrides the command line interface (CLI).

The **WINEVENTLOGS** statement is a comma-delimited list with no spaces that can contain the following values: **Application**, **Directory** (Directory service), **DNS**, **FRS**, **Security**, **System**, **All**, and **None**.

In the following **WINEVENTLOGS** statement, the System, Security, and File Replication service event logs are monitored and all others are ignored:

```
WINEVENTLOGS=System,Security,FRS
```

In the following statement, all event logs are monitored:

```
WINEVENTLOGS=All
```

If a statement contains one or more event logs as well as the **All** or **None** option, the **All** or **None** option is used and the list of event logs is ignored. In the following example, all event logs are monitored even though specific event logs are also listed:

```
WINEVENTLOGS=DNS,Directory,All
```

If a statement contains both the **All** and **None** options, the **None** option overrides all other options. In the following example, no event logs are monitored:

```
WINEVENTLOGS=Application,All,FRS,Directory,None
```

After changing the **WINEVENTLOGS** statement in the **tecad_win.conf** file, you must restart the adapter for the changes to take effect.

Prefiltering Windows Log Events

You can improve Windows event log adapter performance by filtering events in the Windows event logs so only those events that are of importance to administrators are processed by the adapter. This type of filtering is called prefiltering because it specifies selection criteria based on the raw Windows event record rather than the formatted IBM Tivoli Enterprise Console event. The prefiltering is performed before the event is formatted into an IBM Tivoli Enterprise Console event and subjected to any filtering specified with the **Filter** or **FilterCache** configuration file keywords.

Like other adapter filtering, prefiltering is specified in the adapter configuration file using a similar syntax. The prefiltering statements, **PreFilter** and **PreFilterMode**, are described in “Configuration File” on page 112.

As with any modification to an adapter configuration file, you must stop and restart the adapter for the changes to take effect.

There are four attributes of the Windows event logs that you can use in defining prefilter statements. They are described in the following list:

Log Specifies one or more of the Windows event logs to prefilter. Valid values are **System**, **Security**, **Application**, **DNS**, **FRS**, **Directory**, or any combination of these separated by commas. The default is all these event logs.

EventId
Specifies the event number assigned by Windows. You can specify up to sixteen event numbers. Multiple event numbers must be separated by commas.

Source

The source that logged the event to the Windows event log. You can specify up to sixteen sources. Multiple sources must be separated by commas.

EventType

The classification of the event assigned by Windows. Valid values are as follows:

- Error
- Warning
- Information
- AuditSuccess
- AuditFailure
- Unknown

The following examples show prefiltering statements. The first statement is on multiple lines due to space restrictions.

```
PreFilter:Log=Application;Source=MyApp;EventId=1000,2000, \
3000;EventType=Warning,Information;

PreFilter:Log=Security;

PreFilter:Log=Application;Source=TECWinAdapter;
```

Format File

The format file contains message format descriptions and their mappings to BAROC events. The message fields of a Windows event are matched against the format descriptions in this file and when a match succeeds, the corresponding IBM Tivoli Enterprise Console event is generated by the adapter. The format file contains predefined mappings for some common Windows events and can be customized to add any new messages.

A Windows event is written to an ASCII message in the following sequence:

- The date expressed as month, day, time, and year.
- The event category, expressed as an integer.
- The event type (Error, Warning, Information, AuditSuccess, AuditFailure, Unknown).
- The Windows security ID; any spaces in this field are replaced by an underscore if the proper registry variable is set.
- The Windows source; any spaces in this field are replaced by an underscore if the proper registry variable is set.
- The Windows event identifier.
- The message text.

The subfields, except the message text field, are derived from the event header in the Windows event object. The output message after formatting is bound against a format description. A formatted error message from the Windows service control manager can look like the following example:

```
Jan 15 15:06:19 1998 0 Error N/A Service_Control_Manager 7024 \
The UPS service terminated with service-specific error 2481.
```

For details about format files, see “Format File” on page 17 and Appendix B, “Format File Reference” on page 145.

Registry Variables

Registry variables are used to control the operation of the Windows event log adapter. Changes made to registry variables take effect immediately; there is no need to stop and restart the adapter. Use the registry editor (**regedt32**) provided by Windows to view and modify registry variables.

Note: It is not necessary to modify the registry variables for the Windows event log adapter to function. The registry variables are automatically set to the correct default values when the Windows event log adapter is installed.

All of the registry variables for the Windows event log adapter are located in the **\HKEY_LOCAL_MACHINE\SYSTEM \CurrentControlSet\Services\TECWinAdapter** directory. The following are the adapter registry variables:

Note: When you change the registry entries for any registry variable with a name ending with **EventsProcessedTimeStamp**, you must also change the registry entries for the corresponding registry variable with a name ending with **EventsProcessed**. For example, if you change the registry entry for **ApplicationEventsProcessedTimeStamp**, you must also change **ApplicationEventsProcessed**.

If both values are not changed, the adapter ends unexpectedly, the **PollingInterval** criteria are met, and a message similar to the following is sent:

```
msg='TECWinAdapter shuts down.Error: older event on \
ApplicationEventsProcessed : (1,920433843) vs  last processed \
event(1,923673952).';
```

To prevent this, stop the adapter and then make the necessary registry changes. When you restart the adapter, a consistency check updates the registry entry for the appropriate variable ending with **EventsProcessed** to match the correct value based on the corresponding variable ending with **EventsProcessedTimeStamp**.

ApplicationEventsProcessed
Contains the highest event number in the Windows Application Log that the adapter has processed. The adapter uses this variable to keep track of how many events it has read and sent to the event server so that the adapter can start at the next event the next time it polls the log. You can lower the **ApplicationEventsProcessed** variable if you want an event to be read and processed again. To process all messages in the Application Log, set the **ApplicationEventsProcessed** variable to 1.

ApplicationEventsProcessedTimeStamp
Contains the time stamp for the corresponding event identified by the value of the **ApplicationEventsProcessed** variable.

DirectoryEventsProcessed
Contains the highest event number in the Windows active directory server log that the adapter has processed. The adapter uses this variable to keep track of how many events it has read and sent to the event server so that the adapter can start at the next event the next time it polls the log. You can lower the **DirectoryEventsProcessed** variable if you want an event to be read and processed again. To process all messages in the Directory Service Log, set the **DirectoryEventsProcessed** variable to 1.

DirectoryEventsProcessedTimeStamp
Contains the time stamp for the corresponding event identified by the value of the **DirectoryEventsProcessed** variable.

DNSEventsProcessed
Contains the highest event number in the Windows DNS Server Log that the adapter has processed. The adapter uses this variable to keep track of how many events it has read and sent to the event server so that the adapter can start at the next event the next time it polls the log. You can lower the **DNSEventsProcessed** variable if you want an event to be read and processed again. To process all messages in the DNS Server Log, set the **DNSEventsProcessed** variable to 1.

DNSEventsProcessedTimeStamp
Contains the time stamp for the corresponding event identified by the value of the **DNSEventsProcessed** variable.

FileReplicationEventsProcessed
Contains the highest event number in the Windows File Replication service event log that the adapter has processed. The adapter uses this variable to keep track of how many File Replication service log events it has read and sent to the event server so that the adapter can start at the next event the next time it polls the log. You can lower the **FileReplicationEventsProcessed** variable if you want an event to be read and processed again. To process all messages in the File Replication service log, set the **FileReplicationEventsProcessed** variable to 1.

FileReplicationEventsProcessedTimeStamp
Contains the time stamp for the corresponding event identified by the value of the **FileReplicationEventsProcessed** variable.

PollingInterval
The adapter polls the Windows event logs for new events at intervals when it does not receive any events automatically. The **PollingInterval** variable specifies the upper frequency limit, in seconds, to poll the Windows event logs. The default value is 120 seconds.

Polling begins at 5 seconds. If a new event is detected, the next polling frequency begins at 5 seconds again. If no event is detected from a poll, the polling interval is doubled, until the upper limit is reached. After the upper limit is reached, the polling frequency remains at that interval until a new event is detected; then, it is reset to 5 seconds.

Note: If there are buffered events, but no incoming events, the time still doubles until the set **PollingInterval** time. To avoid this, set **PollingInterval** to a lower number. The **PollingInterval** setting is in the registry in **HKEY_LOCAL_MACHINE\SYSTEM\ CurrentControlSet\Services\ TECWinAdapter**. This is not set by default and must be added to the registry to alter the default value of 120 seconds.

SecurityEventsProcessed
Contains the highest event number in the Windows Security Log that the adapter has processed. The adapter uses this variable to keep track of how many events it has read and sent to the event server so that the adapter can start at the next event the next time it polls the log. You can lower the **SecurityEventsProcessed** variable if you want an event to be read and processed again. To process all messages in the Security Log, set the **SecurityEventsProcessed** variable to 1.

SecurityEventsProcessedTimeStamp
Contains the time stamp for the corresponding event identified by the value of the **SecurityEventsProcessed** variable.

SystemEventsProcessed
Contains the highest event number in the Windows event log that the adapter has processed. The adapter uses this variable to keep track of how many log events it has read and sent to the event server so that the adapter can start at the next event the next time it polls the log. You can lower the **SystemEventsProcessed** variable if you want an event to be read and processed again. To process all messages in the event log, set the **SystemEventsProcessed** variable to 1.

SystemEventsProcessedTimeStamp
Contains the time stamp for the corresponding event identified by the value of the **SystemEventsProcessed** variable.

TECInstallPath
Specifies the directory that contains the Windows event log adapter executable files and run-time files. This variable is normally set to *drive*:*adapter_dir*, where *drive* and *adapter_dir* are the drive and directory, respectively, that contain the adapter executable files and run-time files. Only change the **TECInstallPath** variable if you move the adapter executable files and run-time files after you have installed the adapter.

Low Memory Registry Variables

When enabled, this feature checks the amount of available memory before the Windows event log adapter attempts to send an event. If the amount of free memory is extremely low, the Windows event log adapter returns to a suspended state until more memory is available, which prevents the adapter from failing. However, because of the amount of resources this consumes, only enable this feature when available memory is so low that the adapter is failing and you have no other way to solve the problem.

To enable this feature, you must set at least one of following registry variables in the **\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TECWinadapter** registry path:

yellow_alert_limit
When free memory is below this level, the adapter sends a warning that indicates the adapter *might* return to a suspended state until more memory is available and lists the amount of free memory. The default value is 40 Mb.

red_alert_limit
When free memory is below this level, the adapter sends a warning and lists the amount of free memory, then returns to a suspended state for 1 minute. After 1 minute, the adapter checks free memory again; if free memory is still below this level, the adapter returns to a suspended state for another minute and repeats until free memory is higher than this value. The default is 20 Mb.

emergency_memsize
This is the amount of memory the adapter keeps in reserve for low memory situations. When the **red_alert_limit** is reached, the adapter frees this memory to make sure there is enough memory available to send the **red_alert_limit** warning. The default is 2 Mb.

Any values, which you do not set, use the default values when you enable this feature. The adapter only checks these values at startup.

Adapter Administrator Roles for Windows

Both the service and non-service version of TME adapters on Windows run under the local **SYSTEM** account (the built-in Windows account). You must create a Tivoli administrator that grants the Tivoli role of **senior** (or higher) to the **SYSTEM** account so that the adapters can send events to the event server. Otherwise, the TME adapters exit on the first event.

To create a Tivoli administrator with **senior** (or higher) authorization role, do the following:

1. Select **Create Administrators** from the **Administrators** icon.
2. You can leave the **User Login Name** and **Group Login Name** fields blank.
3. Type in **SYSTEM** in the **Set Login Names** dialog.
4. Select **senior** (or higher) in the **Set TMR Roles** dialog.

Starting the Adapter

By default, the adapter is always started when Windows is started. If you are using the Windows service version of the Windows event log adapter, you can use the Windows tools to operate the adapter. For example, you can start and stop the adapter using Windows Control Panel Services. You can also manually start the adapter from the command line with the following command:

```
net start TECWinAdapter
```

Note: The endpoint adapter is automatically started as a step in the adapter installation process when the adapter configuration profile (ACP) is distributed using the Adapter Configuration Facility (ACF).

Stopping the Adapter

You can manually stop the adapter from the command line with the following command:

```
net stop TECWinAdapter
```

Note: The endpoint adapter can be automatically stopped by distributing an ACP that has the adapter start command removed from the after-file-distribution actions. See the *IBM Tivoli Enterprise Console User's Guide* for additional information.

Events Listing

The following table shows the class names and severities of all events defined for the Windows event log adapter. You can use it to get a sense of how Windows events are mapped to IBM Tivoli Enterprise Console events and to determine if you want to make any changes. The events are defined in the BAROC file.

See the *IBM Tivoli Enterprise Console Rule Builder's Guide* for more information about customizing the BAROC file.

Event Class Structure

Event classes are defined hierarchically, with child classes inheriting attribute value defaults from the parent. The Windows event classes follow a simple hierarchy.

The adapter fills in the following attribute default values. The attributes are used in event group filters.

source NT

sub_source
NT

hostname
hostname where the event originated

The following events are defined in BAROC file:

Event Class	Severity
NT_Base	
NT_Base_Event	
NT_Diskfull	WARNING
NT_Share_Dir_Missing	WARNING
NT_Service_Start	WARNING
NT_Service_Stop	WARNING
NT_Out_Of_Paper	WARNING
NT_Printer_Out_Of_Paper	WARNING
NT_Low_Virtual_Memory	WARNING
NT_Security_Db_Not_In_Sync	WARNING
NT_Registry_Bad_DB	WARNING
NT_NCNB_Error	WARNING
NT_Parity_Error	WARNING
NT_Power_Failure	WARNING
NT_Thread_Create_Fail	WARNING
NT_Semaph_Create_Fail	WARNING
NT_Monitor_Start	WARNING
NT_TCPService_Fail	
NT_Master_Browser_Conflict	
NT_Document_Print_Success	
NT_Document_Print_Deleted	
NT_Internal_Error_In_The_DHCP_Server	
NT_Performance_Alert	
NT_Capacity_Alert	
NT_Performance_Monitor	
NT_Trustee_Relationship_Failed	
NT_Service_Started	
NT_Service_Terminated	
NT_Printer_Error	

Event Class	Severity
NT_Printer_Was_Set	
NT_Printer_Was_Created	
NT_Printer_Pending_Deletion	
NT_Security_Database	
NT_Security_Database_Error	
NT_Insight_Agent_Disk_Alert	
NT_DHCP_Rejected_Allocation_Request	
NT_Domain_Not_Contactable	
NT_WINS_Alert	
NT_WINS_Server_Alert	
NT_Master_Browser	
NT_Trustee_Relationship	
NT_Timeserv_Worked	
NT_Timeserv_Failed_1	
NT_Timeserv_Failed_2	
NT_Timeserv_Failed_3	
NT_Timeserv_Failed_4	
NT_Timeserv_Failed_5	
NT_Timeserv_Failed_6	
NT_License_Service_No_License_Available	
NT_License_Service_Out_Of_Licenses	
NT_Restore	
NT_Backup	
NT_Replicator_Did_Not_Send_Update	
NT_Replicator_System_Error	
NT_Replicator	
NT_Tivoli_Courier	
NT_Tivoli_TEC_Adapter	
NT_Tivoli_TEC_Adapter_Error_Sending_Alert	
NT_Sophos_Sweep	
NT_SNMP	
NT_Insight_Manager_Error	
NT_Insight_Manager	
NT_Privileged_Service_Called	
NT_Trusted_Process_Logon_Success	
NT_Logon_Successful	
NT_Logon_Failure	
NT_User_Logoff	
NT_Log_Clear_Successful	
NT_Account_Management_Success	
NT_Group_Management_Change_Success	

Event Class	Severity
NT_Global_Group_Changed	
NT_Local_Group_Member_Removed	
NT_Account_Password_Change_Success	
NT_Server_Start	
NT_Application_Error	
NT_Table_Reached_Maximum_Size	
NT_Handle_Closed	
NT_Object_Open	
NT_Audit_Policy_Change	
NT_Duplicate_Name	WARNING

tecad_win Command

The Windows event log adapter includes the **tecad_win** command, which enables you to start the adapter in non-service mode. The command description is on the following pages.

tecad_win

Starts the Windows event log adapter in non-service mode.

SYNOPSIS

tecad_win.exe [**–d**] [**–c ConfigFile**] [**–L none** | *EventLog* ...]

DESCRIPTION

The **tecad_win** command starts the Windows event log adapter in non-service mode. You can use the non-service mode for diagnostic purposes or to view event messages in a Windows console window. The Windows service mode adapter must be stopped before the non-service mode adapter is started. To stop the service mode adapter, run the following from the command line:

```
net stop TECWinAdapter
```

Before starting the non-service adapter, set the **TECADHOME** environment variable.

Authorization: none

Arguments:

–c *ConfigFile*
: Specifies the configuration file for the Windows event log adapter. If a value is not specified, the **tecad_win.conf** file in the current directory is used. If the **–c** argument is used, you can optionally specify a full path name for the configuration file; otherwise, one of the appropriate directories specified in "File Location" on page 9 is used.

–d
: Shows debug information as events are gathered and transmitted. This argument also selects a verbosity level of 1.

  Note: When running a non-TME version of the Windows event log adapter in this mode, make sure that no other adapters of the same source are running at the same time.

–L
: Specifies which Windows event logs, if any, to monitor.

  none Specifies that no Windows event logs are monitored.

  EventLog ...

  Specifies which Windows event logs are monitored. Values are **ApplicationLog**, **DirectoryLog**, **DNSServerLog**, **FileReplicationLog**, **SecurityLog**, and **SystemLog**. When specifying more than one event log, separate the entries with a space.

EXAMPLES

The following command starts the Windows event log adapter in diagnostic mode:

```
tecad_win –d
```

The following command starts the Windows event log adapter with the **myconfile.conf** configuration file:

```
tecad_win –c myconfile.conf
```

Note: The **.conf** file must be in the **/etc** directory where the adapter is installed.

Troubleshooting the Windows Event Log Adapter

Perform the following steps to troubleshoot the Windows event log adapter:

1. Stop the Windows event log adapter that is currently running by pressing the **Esc** key in the command window session that is running the Windows event log adapter. Pressing the **Ctrl**+**c** key combination in the command window session that is running the Windows event log adapter also stops the adapter.
2. Start the adapter in debug mode:

   tecad_win –**d** –**c** *Config_File*
3. Generate test events and see if the adapter receives them. Do this by starting and stopping a service that logs to the Windows Event Manager. For example, you can use the Windows Control Panel Services to stop the FTP Server and then start it. This adds an event entry in the Windows Security Log that is picked up by the Windows event log adapter.

   Another effective way to generate and monitor Windows events is to run the Windows User Manager application (located in the **Administrative Tools** folder). Select **Audit** from the **Policies** menu and choose from the different activities that Windows can monitor. You want these items to be audited and then picked up by the Windows event log adapter.

   Yet another method is to set up an alert in Windows Performance Monitor (located in the **Administrative Tools** folder) to go off every 30 seconds when the CPU usage is less than 100%.
4. When events arrive, the adapter prints messages to the screen indicating the class and the attribute values in the class.

   If you do not see any messages, the adapter is not receiving events from the Windows event logs.

   For example, you should see a message that the FTP server has registered as a trusted login process. If you do not see this message, run the Windows User Manager application (located in the Administrative Tools folder), select **Audit** from the **Policies** menu and choose **Restart**, **Shutdown**, and **System** events to be audited for **Success** and **Failure**. Then stop and restart the Windows FTP server as described in steps 1 and 2.
5. If you see the messages, the adapter is receiving events and processing them. Run the **wtdumprl** command on the event server and verify that the messages are actually showing up in the reception log. If not, the events were not received by the event server or there is a problem with the event server reception process. Check the adapter configuration file to verify that **ServerLocation** and **ServerPort** are properly defined. If the event class appears in any filter entry in the configuration file, the event is not sent to the event server. The administrator who started the adapter must have the required roles if you are running the TME version of the adapter. For a TME adapter, running the **odstat** command can offer some clues as to what failed.
6. If the reception log has a **PARSING_FAILED** error, the BAROC definition of the class does not match the event that is being received from the adapter. Usually the error messages pinpoint the problem.
7. If the previous steps do not indicate any problem and you do not see the new events in the IBM Tivoli Enterprise Console product, there might be a problem with the event group filters. Make sure the class filters match the classes in the BAROC files.
8. Change all **/dev/null** entries in the **.err** file to the file name you want. Stop and restart the adapter, send an event through, and then look in the trace file to see what processing was done on the event.

Chapter 11. Windows NT Event Log Adapter

The adapter for the Microsoft Windows NT event log forwards events from a Windows NT system to the event server. It is registered with the start-up configuration of Windows NT so that the adapter is started with all the other applications that are automatically started when Windows NT is started.

Note: Only a single instance of the Windows NT or Windows event log adapter can be run on a managed node or endpoint.

The adapter is a WIN32 process that reads events generated on a Windows NT system, formats them according to the specification in the format file, and forwards them using Winsock TCP/IP to an event server for further processing.

Windows NT events are gathered from the three Windows NT event logs (System, Application, and Security) maintained by the Windows NT Event Manager, and from any other ASCII log files residing on the Windows NT system. The Windows NT event log adapter tracks the messages read from the Windows NT event logs using three registry variables that contain the most recent highest message read for the System, Application, and Security logs, whether the Windows NT event log adapter is running continuously or is restarted. You can alter this behavior using the appropriate switches when the Windows NT event log adapter is started.

Two versions of the Windows NT event log adapter are provided. One is built as a Windows NT service, while the other is a WIN32 process that is not a Windows NT service. You should normally run the Windows NT service version, since it runs even when no user is logged in. The non-service version can be used to help you view console messages for diagnostic purposes. Other than the service-related differences, both versions perform identically.

This chapter describes how to configure and start the Windows NT event log adapter.

Adapter Files

The Windows NT event log adapter package consists of the following files:

README The readme file.

tecinstl_nt.cmd
The adapter installation batch file.

instlsrv.exe The adapter installation assist executable file.

tecadnts.exe The adapter service executable file.

tecad_nt.exe The adapter non-service executable file.

tecad_nt.conf The configuration file.

tecad_nt.fmt The format file.

tecad_nt.cds The class definition statement (CDS) file.

tecad_nt.baroc The BAROC file.

postemsg.exe The command line interface program to send an event to an event server.

tecad_nt.err The error file.

Before starting the event server, check the configuration file to determine if it defines the preferred adapter behavior.

Configuration File

The configuration file defines the behavior of the adapter. This file can contain the common keywords described in "Configuration File" on page 9, as well as the following adapter-specific keywords:

HostnameIsAdapterHost
Specifies whether the hostname attribute for Windows NT Event Log events is set to the host on which the adapter is running (the default) or the host where the event originated.

If set to **NO** or **no**, the hostname attribute is set to the **COMPUTER** field from the Windows NT Event Log.

Note: This only applies to events from the Windows NT Event Log, not those generated from log files specified in **LogSources**. Those events always have the hostname attribute set to the host on which the adapter is running.

The **COMPUTER** name returned from the Windows NT Event Log might not be the same as the **ManagedNode** name (which is case-sensitive) of the host where the event originated. You must take this into consideration if you run tasks or programs from the IBM Tivoli Enterprise Console product or the rule base, because they might use the hostname attribute to determine where they run.

LanguageID Sets the language event log messages to be formatted in English or the native language. Valid values are one of the following:

ENGLISH
Messages are formatted in English.

DEFAULT
The adapter attempts to format event log messages in the default language based on the locale value set in Windows NT. If the adapter cannot use the default language, it uses English. The value **DEFAULT** can only be used in languages that have 8-bit wide characters.

The format file is in English. The Windows NT event logs are in your native language. If your native language is not English, you must rewrite the format file in your native language.

LogSources Specifies the ASCII log files to poll for messages. The complete path to each file must be specified, and file names must be separated by commas; no spaces or other separators can be used. A log file source need not exist when the adapter is started; it will be polled when it is created.

If a file truncates while the adapter is active, the adapter automatically resets its internal pointer to the beginning of the file.

If during the polling interval the file is overwritten, removed, or recreated with more lines than the previous poll, only the number of lines greater than the previous line count is read. For example, the file has one line. After the poll interval elapses, the file is overwritten with two lines. Only the second line is read on the next polling.

NumEventsToCatchUp

Specifies which event in the Windows NT event logs that the adapter starts with. This option provides some flexibility if the source being monitored is new or the adapter has been stopped for an extended period of time. Valid values are as follows:

0 Start with the next event in the logs.

–1 Start with the oldest event in the logs.

n *n* represents any number other than zero (0) or –1. Start with the *n*th event from the most current event in the logs; that is, start *n* events back from the most current event in the logs. If *n* is greater than the number of events that are available, all the events that are available are processed.

PollInterval Specifies the frequency, in seconds, to poll each log file listed in the **LogSources** keyword for new messages. The default value is 120 seconds.

Polling begins at 5 seconds. If a new event is detected, the next polling frequency begins at 5 seconds again. If no event is detected from a poll, the polling interval is doubled, until the upper limit is reached. After the upper limit is reached, the polling frequency remains at that interval until a new event is detected; then, it is reset to 5 seconds.

Note: If there are queued events, but no incoming events, the time still doubles until the set polling interval time. To avoid this, set the polling interval to a lower number. The polling interval setting is in the registry in **HKEY_LOCAL_MACHINE\SYSTEM\ CurrentControlSet\Services\ TECNTAdapter**.

PreFilter Specifies how events in a Windows NT event log are filtered before adapter processing. **PreFilter** statements are used by **PreFilterMode** when determining which events are sent from an event log to the adapter. An event matches a **PreFilter** statement when each *attribute=value* specification in the **PreFilter** statement matches an event in the event log. A **PreFilter** statement must contain at least the log specification and can contain up to three additional specifications, which are all optional: event ID, event type, and event source. The order of the attributes in the statement does not matter.

The basic format of the **PreFilter** statement is as follows:

```
PreFilter:Log=log_name;EventId=value;EventType=value;Source=value;
```

You can specify multiple values for each attribute by separating each with a comma.

Each **PreFilter** statement must be on a single line.

The **PreFilter** keyword is optional. All Windows NT log events are sent to the adapter if prefilters are not specified and **PreFilterMode=OUT**.

For additional information about prefiltering Windows NT log events, see "Prefiltering Windows NT Log Events" on page 130.

PreFilterMode

Specifies whether Windows NT log events that match a **PreFilter** statement are sent (**PreFilterMode=IN**) or ignored (**PreFilterMode=OUT**). Valid values are **IN**, **in**, **OUT**, or **out**. The default is **OUT**.

The **PreFilterMode** keyword is optional; if **PreFilterMode** is not specified, only events that do not match any **PreFilter** statements are sent to the adapter.

Note: If you set **PreFilterMode=IN**, make sure you have one or more **PreFilter** statements defined as well.

For additional information about prefiltering Windows NT event log events, see "Prefiltering Windows NT Log Events" on page 130.

SpaceReplacement

When **SpaceReplacement** is **FALSE**, any spaces in the security ID and subsource fields of the event log messages are left unchanged. When **SpaceReplacement** is **TRUE**, any spaces in the security ID and subsource fields of the event log messages are replaced with underscores. Set **SpaceReplacement** to **TRUE** if the format file expects the security ID and subsource fields to be a single word (that is, uses a **%s** format specification for them). The default setting is **TRUE**.

UnmatchLog Specifies a file to log discarded events that cannot be parsed into an IBM Tivoli Enterprise Console event class by the adapter. The discarded events can then be analyzed to determine if modifications are needed to the adapter format file.

Prefiltering Windows NT Log Events

You can improve Windows NT event log adapter performance by filtering events in the Windows NT event logs so only those events that are of importance to administrators are processed by the adapter. This type of filtering is called prefiltering because it specifies selection criteria based on the raw Windows NT event record rather than the formatted IBM Tivoli Enterprise Console event. The prefiltering is performed before the event is formatted into an IBM Tivoli Enterprise Console event and subjected to any filtering specified with the **Filter** or **FilterCache** configuration file keywords.

Like other adapter filtering, prefiltering is specified in the adapter configuration file using a similar syntax. The prefiltering statements, **PreFilter** and **PreFilterMode**, are described in "Configuration File" on page 128.

As with any modification to an adapter configuration file, you must stop and restart the adapter for the changes to take effect.

There are four attributes of the Windows NT event logs that you can use in defining prefilter statements. They are described in the following list:

Log Specifies one or more of the Windows NT event logs to prefilter. Valid values are **System**, **Security**, **Application**, or any combination of these separated by commas. The default is all three event logs.

EventId
Specifies the event number assigned by Windows NT. You can specify up to sixteen event numbers. Multiple event numbers must be separated by commas.

Source
The source that logged the event to the Windows NT event log. You can specify up to sixteen sources. Multiple sources must be separated by commas.

EventType
The classification of the event assigned by Windows NT. Valid values are as follows:

- Error
- Warning
- Information
- AuditSuccess
- AuditFailure
- Unknown

The following examples show prefiltering statements. The first statement is on multiple lines due to space restrictions.

```
PreFilter:Log=Application;Source=MyApp;EventId=1000,2000, \
3000;EventType=Warning,Information;

PreFilter:Log=Security;

PreFilter:Log=Application;Source=TECNTAdapter;
```

Format File

The format file contains message format descriptions and their mapping to BAROC events. The message fields of a Windows NT event are matched against the format descriptions in this file and when a match succeeds, the corresponding event is generated by the adapter. The format file contains predefined mappings for some common Windows NT events and can be customized to add any new messages.

A Windows NT event is written to an ASCII message in the following sequence:

- The date expressed as month, day, time, and year.
- The event category, expressed as an integer.
- The event type (Error, Warning, Information, AuditSuccess, AuditFailure, Unknown).
- The Windows NT security ID; any spaces in this field are replaced by an underscore if the proper registry variable is set.
- The Windows NT source; any spaces in this field are replaced by an underscore if the proper registry variable is set.
- The Windows NT event identifier.
- The message text.

The subfields, except the message text field, are derived from the event header in the Windows NT event object. The output message after formatting is bound

against a format description. A formatted error message from the Windows NT service control manager can look like the following example:

```
Jan 15 15:06:19 1998 0 Error N/A Service_Control_Manager 7024 \
The UPS service terminated with service-specific error 2481.
```

For details about format files, see "Format File" on page 17 and Appendix B, "Format File Reference" on page 145.

Non-English Format Files

Translated format files are provided for the following languages: German, Spanish, Japanese, Korean, and Brazilian Portuguese. These format files are located in the **de**, **es**, **ja**, **ko**, and **pt_Br** subdirectories of the **TME/TEC/samples/adapter_format_files** directory. Use these files as a starting point for creating language-specific format files. For other supported languages, the default format file is the English version.

Registry Variables

Registry variables are used to control the operation of the Windows NT event log adapter. Changes made to registry variables take effect immediately; there is no need to stop and restart the adapter. Use the registry editor (**regedt32**) provided by Windows NT to view and modify registry variables.

Note: It is not necessary to modify the registry variables for the Windows NT event log adapter to function. The registry variables are automatically set to the correct default values when the Windows NT event log adapter is installed.

All of the registry variables for the Windows NT event log adapter are located in the **\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TECNTAdapter** directory. The following are the adapter registry variables:

Note: When you change the registry entries for any registry variable with a name ending with **EventsProcessedTimeStamp**, you must also change the registry entries for the corresponding registry variable with a name ending with **EventsProcessed**. For example, if you change the registry entry for **ApplicationEventsProcessedTimeStamp**, you must also change **ApplicationEventsProcessed**. This also applies to **SecurityEventsProcessedTimeStamp** and **SecurityEventsProcessed**, and **SystemEventsProcessedTimeStamp** and **SystemEventsProcessed**.

If both values are not changed, the adapter ends unexpectedly, the **PollingInterval** criteria are met, and a message similar to the following is sent:

```
msg='TECNTAdapter shuts down.Error: older event on \
ApplicationEventsProcessed : (1,920433843) vs  last processed \
event(1,923673952).';
```

To prevent this, stop the adapter and then make the necessary registry changes. When you restart the adapter, a consistency check updates the registry entry for the appropriate variable ending with **EventsProcessed** to match the correct value based on the corresponding variable ending with **EventsProcessedTimeStamp**.

ApplicationEventsProcessed
Contains the highest event number in the Windows NT Application Log

that the adapter has processed. The adapter uses this variable to keep track of how many events it has read and sent to the event server so that the adapter can start at the next event the next time it polls the log. You can lower the **ApplicationEventsProcessed** variable if you want an event to be read and processed again. To process all messages in the Application Log, set the **ApplicationEventsProcessed** variable to 1.

ApplicationEventsProcessedTimeStamp
Contains the time stamp for the corresponding event identified by the value of the **ApplicationEventsProcessed** variable.

PollingInterval
The adapter polls the Windows NT event logs for new events at intervals when it does not receive any events automatically. The **PollingInterval** variable specifies the upper frequency limit, in seconds, to poll the Windows NT event logs. The default value is 120 seconds.

Polling begins at 5 seconds. If a new event is detected, the next polling frequency begins at 5 seconds again. If no event is detected from a poll, the polling interval is doubled, until the upper limit is reached. After the upper limit is reached, the polling frequency remains at that interval until a new event is detected; then, it is reset to 5 seconds.

Note: If there are buffered events, but no incoming events, the time still doubles until the set **PollingInterval** time. To avoid this, set **PollingInterval** to a lower number. The **PollingInterval** setting is in the registry in **HKEY_LOCAL_MACHINE\SYSTEM\ CurrentControlSet\Services\ TECNTAdapter**. This is not set by default and must be added to the registry to alter the default value of 120 seconds.

SecurityEventsProcessed
Contains the highest event number in the Windows NT Security Log that the adapter has processed. The adapter uses this variable to keep track of how many events it has read and sent to the event server so that the adapter can start at the next event the next time it polls the log. You can lower the **SecurityEventsProcessed** variable if you want an event to be read and processed again. To process all messages in the Security Log, set the **SecurityEventsProcessed** variable to 1.

SecurityEventsProcessedTimeStamp
Contains the time stamp for the corresponding event identified by the value of the **SecurityEventsProcessed** variable.

SystemEventsProcessed
Contains the highest event number in the Windows NT event log that the adapter has processed. The adapter uses this variable to keep track of how many log events it has read and sent to the event server so that the adapter can start at the next event the next time it polls the log. You can lower the **SystemEventsProcessed** variable if you want an event to be read and processed again. To process all messages in the event log, set the **SystemEventsProcessed** variable to 1.

SystemEventsProcessedTimeStamp
Contains the time stamp for the corresponding event identified by the value of the **SystemEventsProcessed** variable.

TECInstallPath
Specifies the directory that contains the Windows NT event log adapter executable files and run-time files. This variable is normally set to

drive:*adapter_dir*, where *drive* and *adapter_dir* are the drive and directory, respectively, that contain the adapter executable files and run-time files. Only change the **TECInstallPath** variable if you move the adapter executable files and run-time files after you have installed the adapter.

Low Memory Registry Variables

When enabled, this feature checks the amount of available memory before the adapter attempts to send an event. If the amount of free memory is extremely low, the adapter returns to a suspended state until more memory is available, which prevents the adapter from failing. However, because of the amount of resources this consumes, only enable this feature when available memory is so low that the adapter is failing and you have no other way to solve the problem.

To enable this feature, you must set at least one of following registry variables in the **\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TECNTAdapter** registry path:

yellow_alert_limit
When free memory is below this level, the adapter sends a warning that indicates the adapter *might* return to a suspended state until more memory is available and lists the amount of free memory. The default value is 40 Mb.

red_alert_limit
When free memory is below this level, the adapter sends a warning and lists the amount of free memory, then returns to a suspended state for 1 minute. After 1 minute, the adapter checks free memory again; if free memory is still below this level, the adapter returns to a suspended state for another minute and repeats until free memory is higher than this value. The default is 20 Mb.

emergency_memsize
This is the amount of memory the adapter keeps in reserve for low memory situations. When the **red_alert_limit** is reached, the adapter frees this memory to make sure there is enough memory available to send the **red_alert_limit** warning. The default is 2 Mb.

Any values, which you do not set, use the default values when you enable this feature. The adapter only checks these values at startup.

Adapter Administrator Roles for Windows NT

Both the service and non-service version of TME adapters on Windows NT run under the local **SYSTEM** account (the built-in Windows NT account). You must create a Tivoli administrator that grants the Tivoli role of **senior** (or higher) to the **SYSTEM** account so that the adapters can send events to the event server. Otherwise, the TME adapters exit on the first event.

To create a Tivoli administrator with **senior** (or higher) authorization role, do the following:

1. Select **Create Administrators** from the **Administrators** icon.
2. You can leave the **User Login Name** and **Group Login Name** fields blank.
3. Type in **SYSTEM** in the **Set Login Names** dialog.
4. Select **senior** (or higher) in the **Set TMR Roles** dialog.

Starting the Adapter

By default, the adapter is always started when Windows NT is started. If you are using the Windows NT service version of the Windows NT event log adapter, you can use the Windows NT tools to operate the adapter. For example, you can start and stop the adapter using Windows NT Control Panel Services. You can also manually start the adapter from the command line with the following command:

```
net start TECNTAdapter
```

Note: The endpoint adapter is automatically started as a step in the adapter installation process when the adapter configuration profile (ACP) is distributed using the Adapter Configuration Facility (ACF).

Stopping the Adapter

You can manually stop the adapter from the command line with the following command:

```
net stop TECNTAdapter
```

Note: The endpoint adapter can be automatically stopped by distributing an ACP that has the adapter start command removed from the after-file-distribution actions. See the *IBM Tivoli Enterprise Console User's Guide* for additional information.

Events Listing

The following table shows the class names and severities of all events defined for the Windows NT event log adapter. You can use it to get a sense of how Windows NT events are mapped to IBM Tivoli Enterprise Console events and to determine if you want to make any changes. The events are defined in the BAROC file.

See the *IBM Tivoli Enterprise Console Rule Builder's Guide* for more information about customizing the BAROC file.

Event Class Structure

Event classes are defined hierarchically, with child classes inheriting attribute value defaults from the parent. The Windows NT event classes follow a simple hierarchy.

The adapter fills in the following attribute default values. The attributes are used in event group filters.

source NT

sub_source
NT

hostname
hostname where the event originated

The following events are defined in BAROC file:

Event Class	Default Severity
NT_Base	
NT_Base_Event	
NT_Diskfull	WARNING

Event Class	Default Severity
NT_Share_Dir_Missing	WARNING
NT_Service_Start	WARNING
NT_Service_Stop	WARNING
NT_Out_Of_Paper	WARNING
NT_Printer_Out_Of_Paper	WARNING
NT_Low_Virtual_Memory	WARNING
NT_Security_Db_Not_In_Sync	WARNING
NT_Registry_Bad_DB	WARNING
NT_NCNB_Error	WARNING
NT_Parity_Error	WARNING
NT_Power_Failure	WARNING
NT_Thread_Create_Fail	WARNING
NT_Semaph_Create_Fail	WARNING
NT_Monitor_Start	WARNING
NT_TCPService_Fail	
NT_Master_Browser_Conflict	
NT_Document_Print_Success	
NT_Document_Print_Deleted	
NT_Internal_Error_In_The_DHCP_Server	
NT_Performance_Alert	
NT_Capacity_Alert	
NT_Performance_Monitor	
NT_Trustee_Relationship_Failed	
NT_Service_Started	
NT_Service_Terminated	
NT_Printer_Error	
NT_Printer_Was_Set	
NT_Printer_Was_Created	
NT_Printer_Pending_Deletion	
NT_Security_Database	
NT_Security_Database_Error	
NT_Insight_Agent_Disk_Alert	
NT_DHCP_Rejected_Allocation_Request	
NT_Domain_Not_Contactable	
NT_WINS_Alert	
NT_WINS_Server_Alert	
NT_Master_Browser	
NT_Trustee_Relationship	
NT_Timeserv_Worked	
NT_Timeserv_Failed_1	
NT_Timeserv_Failed_2	

Event Class	Default Severity
NT_Timeserv_Failed_3	
NT_Timeserv_Failed_4	
NT_Timeserv_Failed_5	
NT_Timeserv_Failed_6	
NT_License_Service_No_License_Available	
NT_License_Service_Out_Of_Licenses	
NT_Restore	
NT_Backup	
NT_Replicator_Did_Not_Send_Update	
NT_Replicator_System_Error	
NT_Replicator	
NT_Tivoli_Courier	
NT_Tivoli_TEC_Adapter	
NT_Tivoli_TEC_Adapter_Error_Sending_Alert	
NT_Sophos_Sweep	
NT_SNMP	
NT_Insight_Manager_Error	
NT_Insight_Manager	
NT_Privileged_Service_Called	
NT_Trusted_Process_Logon_Success	
NT_Logon_Successful	
NT_Logon_Failure	
NT_User_Logoff	
NT_Log_Clear_Successful	
NT_Account_Management_Success	
NT_Group_Management_Change_Success	
NT_Global_Group_Changed	
NT_Local_Group_Member_Removed	
NT_Account_Password_Change_Success	
NT_Server_Start	
NT_Application_Error	
NT_Table_Reached_Maximum_Size	
NT_Handle_Closed	
NT_Object_Open	
NT_Audit_Policy_Change	
NT_Duplicate_Name	WARNING

tecad_nt Command

The Windows NT event log adapter includes the **tecad_nt** command, which enables you to start the adapter in non-service mode. The command description is on the following pages.

tecad_nt

Starts the Windows NT event log adapter in non-service mode.

SYNOPSIS

tecad_nt.exe [**–d**] [**–c ConfigFile**] **[–L none** | *EventLog* ...]

DESCRIPTION

The **tecad_nt** command starts the Windows NT event log adapter in non-service mode. You can use the non-service mode for diagnostic purposes or to view event messages in a Windows NT console window. The Windows NT service mode adapter must be stopped before the non-service mode adapter is started. To stop the service mode adapter, run the following from the command line:

```
net stop TECNTAdapter
```

Before starting the non-service adapter, set the **TECADHOME** environment variable.

Authorization: none

Arguments:

–c *ConfigFile*
: Specifies the configuration file for the Windows NT event log adapter. If a value is not specified, the **tecad_nt.conf** file in the current directory is used. If the **–c** argument is used, you can optionally specify a full path name for the configuration file; otherwise, one of the appropriate directories specified in "File Location" on page 9 is used.

–d
: Shows debug information as events are gathered and transmitted. This argument also selects a verbosity level of 1.

  Note: When running a non-TME version of the Windows NT event log adapter in this mode, make sure that no other adapters of the same source are running at the same time.

–L
: Specifies which Windows NT event logs, if any, to monitor.

  none Specifies that no Windows NT event logs are monitored.

  EventLog ...
  : Specifies which Windows NT event logs are monitored. Values are **ApplicationLog**, **SecurityLog**, and **SystemLog**. When specifying more than one event log, separate the entries with a space.

EXAMPLES

The following command starts the Windows NT event log adapter in diagnostic mode:

```
tecad_nt -d
```

The following command starts the Windows NT event log adapter with the **myconfile.conf** configuration file:

```
tecad_nt -c myconfile.conf
```

Note: The **.conf** file must be in the **/etc** directory where the adapter is installed.

Troubleshooting the Windows NT Event Log Adapter

Perform the following steps to troubleshoot the Windows NT event log adapter:

1. Stop any Windows NT event log adapters that are currently running by pressing the **Esc** key in the command window session that is running the Windows NT event log adapter. Pressing the **Ctrl**+**c** key combination in the command window session that is running the Windows NT event log adapter also stops the adapter.
2. Start the adapter in debug mode:

   tecad_nt –**d** –**c** *Config_File*
3. Generate test events and see if the adapter receives them. Do this by starting and stopping a service that logs to the Windows NT Event Manager. For example, you can use Windows NT Control Panel Services to stop the FTP Server and then start it. This adds an event entry in Windows NT Security Log that is picked up by the Windows NT event log adapter.

   Another effective way to generate and monitor Windows NT events is to run Windows NT User Manager application (located in the **Administrative Tools** folder). Select **Audit** from the **Policies** menu and choose from the different activities that Windows NT can monitor. You want these items to be audited and then picked up by the Windows NT event log adapter.

   Yet another method is to set up an alert in Windows NT Performance Monitor (located in the **Administrative Tools** folder) to go off every 30 seconds when the CPU usage is less than 100%.
4. When events arrive, the adapter prints messages to the screen indicating the class and the attribute values in the class.

   If you do not see any messages, the adapter is not receiving events from the Windows NT event logs.

   For example, you should see a message that the FTP server has registered as a trusted login process. If you do not see this message, run Windows NT User Manager application (located in the Administrative Tools folder), select **Audit** from the **Policies** menu and choose **Restart**, **Shutdown**, and **System** events to be audited for **Success** and **Failure**. Then stop and restart the Windows NT FTP server as described in steps 1 and 2.
5. If you see the messages, the adapter is receiving events and processing them. Run the **wtdumprl** command on the event server and verify that the messages are actually showing up in the reception log. If not, the events were not received by the event server or there is a problem with the event server reception process. Check the adapter configuration file to verify that **ServerLocation** and **ServerPort** are properly defined. If the event class appears in any filter entry in the configuration file, it will not be sent to the event server. The administrator who started the adapter must have the required roles if you are running the TME version of the adapter. For a TME adapter, running the **odstat** command can offer some clues as to what failed.
6. If the reception log has a **PARSING_FAILED** error, the BAROC definition of the class does not match the event that is being received from the adapter. Usually the error messages pinpoint the problem.
7. If the previous steps do not indicate any problem and you do not see the new events in the IBM Tivoli Enterprise Console product, there might be a problem with the event group filters. Make sure the class filters match the classes in the BAROC files.
8. Change all **/dev/null** entries in the **.err** file to the file name you want. Stop and restart the adapter, send an event through, and then look in the trace file to see what processing was done on the event.

Shutting down the service version of the Windows NT event log adapter can take up to 10 minutes, if the adapter and the CPU are under a heavy load. This delay occurs because the adapter attempts to finish processing all pending events before exiting. The adapter should shut down immediately under normal load conditions.

Appendix A. Files Shipped with Adapters

Notes:

1. The NetView for OS/390® adapters are delivered with Tivoli NetView for OS/390 as part of the Event/Automation Service. Although these adapters are shipped as part of that product, the BAROC files and rule files for them are shipped with the IBM Tivoli Enterprise Console product. For information about additional files shipped with these adapters, see the Tivoli NetView for OS/390 documentation.

The following table lists some of the files used with the shipped adapters. An x indicates the file is used by an adapter.

File	Extension	Adapter								
		AS/400 Alert	**AS/400 Message**	**NetWare**	**OpenView**	**OS/2**	**SNMP**	**UNIX Log File**	**Windows Event Log**	**Windows NT Event Log**
BAROC	**.baroc**	x	x	x	x	x	x	x	x	x
Class definition statement (CDS)	**.cds**	x	x	x	x	x	x	x	x	x
Configuration	**.conf** [1]	x	x	x	x	x	x	x	x	x
Error	**.err**			x	x	x	x	x	x	x
Format	**.fmt**					x		x	x	x
Installation script	**.cfg**[2]				x	x	x	x		
Object identifier	**.oid**				x		x			
Registration	**.lrf**				x					
Rules	**.rls**[3]	x	x		x			x		

1. The AS/400 adapters use a **.mbr** extension.
2. The OS/2 adapter actually uses a command file (**.cmd**) for performing this function.
3. A rules file is not shipped with the AS/400 message adapter. You can create a rules file if needed.

The following table lists the file names for some of the more significant files used for the IBM Tivoli Enterprise Console adapters:

Adapter	Extension	File Name
AS/400 alert	**.baroc**	**/QSYS.LIB/QUSRSYS.LIB/CFG_ALERT.FILE/ ALRBRC.MBR tecad_snaevent.baroc (on event server)**
	.cds	**/QSYS.LIB/QUSRSYS.LIB/CFG_ALERT.FILE/ ALRCDS.MBR**
	.conf	**/QSYS.LIB/QUSRSYS.LIB/CFG_ALERT.FILE/ ALRCFG.MBR**
	.rls	**/QSYS.LIB/QUSRSYS.LIB/CFG_ALERT.FILE/ ALRRLS.MBR tecad_snaevent.rls (on the event server)**
AS/400 message	**.baroc**	**/QSYS.LIB/QUSRSYS.LIB/CFG_MSG.FILE/ MSGBRC.MBR as400msg.baroc (on the event server)**
	.cds	**/QSYS.LIB/QUSRSYS.LIB/CFG_MSG.FILE/ MSGCDS.MBR**
	.conf	**/QSYS.LIB/QUSRSYS.LIB/CFG_MSG.FILE/ MSGCFG.MBR**
NetWare	**.brc**	**tecadnw4.brc**
	.cds	**tecadnw4.cds**
	.cnf	**tecadnw4.cnf**
	.err	**tecadnw4.err**
	.fmt	**tecadnw4.fmt**
OpenView	**.baroc**	**tecad_hpov.baroc**
	.cds	**tecad_hpov.cds**
	.cfg	**tecad_hpov.cfg**
	.conf	**tecad_hpov.conf**
	.err	**tecad_hpov.err**
	.oid	**tecad_hpov.oid**
	.rls	**ov_default.rls**
OS/2	**.baroc**	**tecados2.baroc**
	.cds	**tecados2.cds**
	.cmd	**tecados2.cmd**
	.conf	**tecados2.conf**
	.err	**tecados2.err**
	.fmt	**tecados2.fmt**
SNMP	**.baroc**	**tecad_snmp.baroc**
	.cds	**tecad_snmp.cds**
	.cfg	**tecad_snmp.cfg**
	.conf	**tecad_snmp.conf**
	.err	**tecad_snmp.err**
	.oid	**tecad_snmp.oid**

Adapter	Extension	File Name
UNIX log file	**.baroc**	**tecad_logfile.baroc**
	.cds	**tecad_logfile.cds**
	.cfg	**tecad_logfile.cfg**
	.conf	**tecad_logfile.conf**
	.err	**tecad_logfile.err**
	.fmt	**tecad_logfile.fmt**
	.rls	**log_default.rls**
Microsoft Windows event log	**.baroc**	**tecad_win.baroc**
	.cds	**tecad_win.cds**
	.conf	**tecad_win.conf**
	.err	**tecad_win.err**
	.fmt	**tecad_win.fmt**
Windows NT event log	**.baroc**	**tecad_nt.baroc**
	.cds	**tecad_nt.cds**
	.conf	**tecad_nt.conf**
	.err	**tecad_nt.err**
	.fmt	**tecad_nt.fmt**

Appendix B. Format File Reference

This appendix contains details about format files.

The format file usually has an extension of **.fmt**; see each specific adapter chapter for exact file names. To use non-English characters in a format string, you must enter the non-English characters in the local encodings.

Notes:

1. Although this section discusses the manual text editing of a format file and the file organization, you can accomplish the same results for TME adapters with the Log File Format Editor of the Adapter Configuration Facility (ACF). See the *IBM Tivoli Enterprise Console User's Guide* for information about using the Log File Format Editor.
2. The UNIX log file adapter, NetWare log file, and OS/2 adapter format files are in English only. The Microsoft Windows NT event log format file is in English and localized into a sample file for the Tivoli supported languages. If you have a source that issues events in a non-English language and you are monitoring that source with an adapter that uses a format file, and the format file has not been localized, you must localize the format file in that language.

Format File Location

An English-language format file is located in each of the language subdirectories that are in the same directory as the adapter configuration file. The language subdirectories are as follows:

Language	Subdirectory
English	**/C**
German	**/de**
Spanish	**/es**
French	**/fr**
Italian	**/it**
Japanese	**/ja**
Korean	**/ko**
Brazilian Portuguese	**/pt_BR**
Simplified Chinese	**/zh_CN**
Traditional Chinese	**/zh_TW**

See "File Location" on page 9 for more details.

Format Specifications

The format file is made up of one or more format specifications. A format specification has the following parts:

- Format header

  The keyword **FORMAT** followed by the event class name. This is optionally followed by the **FOLLOWS** keyword and a previously defined class name, as shown in the following example:

  ```
  FORMAT NT_Share_Dir_Missing FOLLOWS NT_Base
  ```

  Note: A format specification with the same class name can be defined more than once. Be careful of using multiply-defined format specification class names with the **FOLLOWS** keyword. Since there is no way to specify which actual format specification is intended, the last one defined in the file that matches the class name is used.

- Format content

  A format string optionally followed by a list of mappings, as shown in the following example:

  ```
  %t %s %s %s %s %s %s The server service was unable to recreate
  the share %s because the directory %s no longer exists.
  sharename $8
  directoryname $9
  ```

- The **END** keyword completes the format specification.

The format header, format string, each mapping, and the **END** keyword must each begin on a new line, as shown in the following example:

```
FORMAT NT_Share_Dir_Missing FOLLOWS NT_Base
%t %s %s %s %s %s %s The server service was unable to recreate
the share %s because the directory %s no longer exists.
sharename $8
directoryname $9
END
```

The **FOLLOWS** relationship is used to allow specific format specifications to be built from generic format specifications using inheritance. When format B follows format A, B inherits all of the mappings (but not the format string) from A. Format B can define any additional mappings, but any mappings redefined by B are *not* inherited from A; that is, format B can override inherited mappings by redefining them.

System log messages typically have a common format consisting of a time stamp, a host name, and event text. These system log message components are represented in a format string using a component-specifier notation very similar to the **printf()** notation used in the C programming language. The following format string describes the entire class of system log messages produced by the UNIX **syslogd** daemon:

```
%t %s %s*
```

System log messages are tokenized into constants and white space. A constant is any consecutive string of non-white spaces. The component specifiers allow the constants and white space to be grouped into more complex tokens when trying to match a format string with a specific message. The component specifiers always end in a constant and not white space. The component specifiers are as follows:

- **%**[*length*]**s**

Matches one constant in the message. The optional length is a decimal number of any size and allows the constant to be truncated to the length if the constant actual length is greater than the specifier length.

- **%**[*length*]**s***

  Matches zero or more constants in the system log message. The optional length is a decimal number of any size and allows any of the accumulated constants to be truncated to the length if the constant actual length is greater than the specifier length.

- **%**[*length*]**s+**

  Matches one or more constants in the message. The optional length is a decimal number of any size and allows any of the accumulated constants to be truncated to the length if the constant actual length is greater than the specifier length.

- **%t**

  Matches a time stamp of the following form:

  ```
  month date time
  ```

Log File Example

The following successful **su** message from a system log is an example of matching a system log message to the generic format specification mentioned in the preceding section:

```
Sep 13 12:17:11 elcap su: 'su root' succeeded for tjones on /dev/ttyp0
```

The component specifiers and matches are as follows:

%t Sep 13 12:17:11

%s elcap

%s* su: 'su root' succeeded for tjones on /dev/ttyp0

The system log message contains some constant parts and some variable parts. The constant parts of the system log message will be the same for *any* successful **su** message. The constant parts are as follows:

- su: 'su
- ' succeeded for
- on

The variable parts of the example system log message are as follows:

- Sep 13 12:17:11
- elcap
- root
- tjones
- /dev/ttyp0

The following example shows how the variable data differs in another successful **su** message:

```
Sep 29 14:57:28 aspen su: 'su root' succeeded for jsmith on /dev/ttypd
```

The general format specification **%t %s %s*** can be specialized for the **Su_Success** event class as follows:

```
%t %s su: 'su %s' succeeded for %s on %s
```

Using the system log message from the preceding September 29 example, the component specifiers and matches are as follows:

%t Sep 29 14:57:28

%s aspen

su: 'su su: 'su

%s root

succeeded for
succeeded for

%s jsmith

on on

%s /dev/ttypd

The white space characters that separate the words of a system log message must also be present in the format string. A single space character (that is, one blank) in the format string will match any number of white space characters in the message. For example, if the space between the colon (:) and the quotation mark (') is deleted in the preceding specialized format string, as shown in the following example, the system log message would no longer match it.

```
%t %s su:'su %s' succeeded for %s on %s
```

Care should be taken when using the arbitrary length repeater component specifiers (**%s*** and ***s+**). The following format string does not make much sense:

```
This is not a good format %s* %s*
```

The first **%s*** matches everything through the end of the message, and the second **%s*** never matches anything. It might appear that this does not matter, but the importance is apparent as discussed in "Mappings" on page 149.

The following format string, however, is meaningful:

```
This is a good format %s* : %s*
```

The first **%s*** matches everything up to the first colon (:), and the second **%s*** now matches everything through the end of the message.

The format string must also reflect whether white space precedes a constant or component specifier. In the following example, both messages match a format string of **%s*company_xyz** because they are preceded by zero (0) or more constants and no white space.

```
company_xyz is logging messages
Acompany_xyz is logging messages
```

However, the following example requires a format string with a space after the **%s*** component specifier, as in **%s* company_xyz**, because it is preceded by white space and does not match the previous format string.

```
the company_xyz is logging messages
```

From the preceding examples, you can see that you can specialize a generic format string to match a more specific event by either replacing component specifiers with constants or by restricting the arbitrary length repeater specifiers to a fixed length, using constants to complete the specifier.

Windows NT Example

The following example is a Windows NT message:

```
Jan 15 15:06:19 1998 0 Error N/A Service_Control_Manager 7024 \
The UPS service terminated with service-specific error 2481.
```

The variable parts are the time stamp (Jan 15 15:06:19 1998), possibly the security ID (N/A), the event ID (7024), the service name (UPS), and the error code (2481). Another system log message uses the same general format, as shown in the following example:

```
Sep 29 14:57:28 1998 0 Error N/A Service_Control_Manager 7025 \
The SNMP service terminated with service-specific error 2482.
```

The constant parts of a system log message are defined by simply embedding them in the format string itself. The variable parts are defined using the component specifier. The format string for the preceding September 29 example could be written as follows:

```
%t %s %s Error %s Service_Control_Manager %s The %s \
service terminated with service-specific error %s.
```

The white space characters that separate the words of a system log message must also be present in the format string. A single space character (that is, one blank) in the format string will match any number of white space characters in the message.

Care should be taken when using the arbitrary length repeater component specifiers (**%s*** and ***s+**). The following format specification does not make much sense:

```
This is not a good format %s* %s*
```

The first **%s*** matches everything through the end of the message, and the second **%s*** never matches anything. It might appear that this does not matter, but the importance becomes apparent as discussed in “Mappings” on page 149.

The following format string, however, is meaningful:

```
This is a good format %s* : %s*
```

The first **%s*** matches everything up to the first colon (:), and the second **%s*** now matches everything through the end of the message.

Mappings

The log file adapters translate system log messages into event class instances containing attribute *name=value* pairs. The event is then sent to the event server. An associated BAROC file containing class definitions at the event server is used to validate the incoming event before processing the event further.

For the log file adapters, the event class for a system log message is determined at the source by matching a system log message to a format string in the format file. After a class is determined by this matching, values must be assigned to the attributes. Attribute values can come from a variety of sources, such as from the system log message itself, from default values provided by the adapter, or from mappings within the format specification of a class in the format file. This section discusses how the mappings in a format specification assign values to attributes.

The mapping part of a format specification consists of zero or more lines that contain a BAROC file attribute name followed by a value specifier. The value specifiers can be one of the following types:

$*i* Where *i* indicates the position of a component specifier in a format string. Each component specifier is numbered from 1 to the maximum number of component specifiers in the format string. For example, in the specialized format specification for the **Su_Success** event shown following, the third **%s** component specifier (in bold) would be referred to in any mappings as **$4**.

```
%t %s su: 'su %s' succeeded for %s on %s
```

The value of a **$*i*** value specifier (also referred to as a variable) is the portion of the system log message that was consumed by the component specifier.

string constant
The value of the attribute is the specified string. If the string is a single constant, it can be specified without surrounding double quotation marks (″ ″); otherwise, double quotation marks must be used.

PRINTF statement
Creates more complex attribute values from other attribute values. The **PRINTF** statement consists of the keyword **PRINTF** followed by a **printf()** C-style format string and one or more attribute names. The format string only supports the **%s** component specifier. The values of the attributes that are used in the **PRINTF** statement must also have been derived from either a **$*i*** value specification or a constant string value specification (they cannot be derived from another **PRINTF** statement). The value of the argument attributes will be used to compose a new constant string according to the format string. This new constant string becomes the value of the attribute.

The following example shows how the **msg** attribute is assigned the constant string value of `date set by mfoster`. User ID `mfoster` was derived from the value assigned to the **set_by** attribute.

```
msg PRINTF("date set by %s", set_by)
```

DEFAULT keyword
Indicates the adapter uses its internal logic to assign a value to the indicated attribute. For example, the UNIX **syslogd** messages contain the host name where the message was logged; the adapter can use this name to derive the **origin** attribute (the protocol address or host name of the originating host).

Note: Adding new **DEFAULT** mappings also requires changes to an adapter source code to add new logic for obtaining attribute values.

Because **DEFAULT** is a keyword, a constant mapping whose value is the string **DEFAULT** must be specified in double quotation marks (″ ″).

LABEL keyword
Indicates the type of machine on which the adapter is running, which provides better control over the **hostname** attribute coming from the adapter. For a managed node, the value is the managed node name; in an endpoint, it is the endpoint name, which is listed in **last.cfg** as **lcs.machine_name**. In a non-TME adapter, the value is the host name of the machine.

Additional Mapping Considerations

Specify only one mapping for each BAROC file attribute.

A mapping can be inherited from a more generic format specification (using the **FOLLOWS** keyword) or can be explicitly defined on the format specification that directly matches the message.

Because the adapter does not access the BAROC file, which resides on the event server, care must be taken to make sure that the format specifications agree with the corresponding BAROC file definitions. If an attribute name is misspelled in a mapping, the adapter will not report an error and will send the event to the event server as usual; however, the event will be discarded by the event server because it does not exactly match a class definition.

There can be attributes in the system log message that do not directly correspond to any BAROC file attributes because the adapter might need to use these values to compose **PRINTF** style constant strings for assigning to attributes. This type of data needs to be assigned to temporary attributes that do not get sent to the event server, but are used in the **PRINTF** statement. Temporary attributes are designated with a hyphen (-) immediately preceding the attribute name in a mapping.

In order to illustrate the use of mappings in format specifications, a sample from the default **tecad_logfile.fmt** file is shown following with a few additions.

```
FORMAT Logfile_Base
%t %s %s*
date $1
hostname $2
msg $3
origin DEFAULT
END

/* login */
// NOTE -- anything enclosed in '/*' and '*/' pairs is considered to
// be a comment.  These comments can extend across multiple lines.
// Anything following a '//' is also considered to be a comment;
// this comment only extends to the end of the line.

FORMAT Logfile_Login FOLLOWS Logfile_Base
%t %s login: %s*
sub_source login
END

FORMAT Root_Login FOLLOWS Logfile_Login
%t %s login: ROOT LOGIN %s*
END

FORMAT Root_Login_Success FOLLOWS Root_Login
%t %s login: ROOT LOGIN %s
on_tty $3
msg PRINTF("root login %s", on_tty)
END

FORMAT Root_Login_Success_From FOLLOWS Root_Login_Success
%t %s login: ROOT LOGIN %s FROM %s
from_host $4
-extra ", with extra stuff!"
msg PRINTF("root login from %s%s", from_host, extra)
END
```

Now, assume that the following system log message is received by the log file adapter:

```
Dec 10 09:45:06 sawmill login: ROOT LOGIN ttyp6 FROM oak
```

The log file adapter will attempt to match this system log message to the most specific format specification. In this case, the event matches the **Root_Login_Success_From** format specification. The event created by the log file adapter will therefore have an event class of **Root_Login_Success_From**. The following mappings then take place:

Mapping Assignments	Source of Mapping
`$1="Dec 10 09:45:06"`	From the **%t** component specification
`$2="sawmill"`	From the first **%s** component specification
`$3="ttyp6"`	From the second **%s** component specification
`$4="oak"`	From the third **%s** component specification
`date="Dec 10 09:45:06"`	From **$1**
`hostname="sawmill"`	From **$2**
`origin= 9.37.43.12"`	From the default value of the origin attribute, as derived by the log file adapter
`sub_source="login"`	From the constant string
`on_tty="ttyp6"`	From **$3**
`from_host="oak"`	From **$4**
`-extra=", with extra stuff!"`	From the constant string
`msg="root login from oak, with extra stuff!"`	From the **PRINTF** statement

The following list describes how values were assigned:

- The **date** and **hostname** attributes were inherited from the **Logfile_Base** class (through the **Logfile_Login**, **Root_Login**, and **Root_Login_Success** classes).
- The **origin** attribute was also inherited from the **Logfile_Base** class, and was assigned the adapter default.
- The **msg** attribute was not inherited from the **Logfile_Base** class, because it was overridden by the **Root_Login_Success_From** class.
- The **sub_source** attribute was inherited from the constant string defined in the **Logfile_Login** class.
- The **on_tty** attribute was inherited from the **Root_Login_Success** class.
- The **from_host** attribute was explicitly defined on the **Root_Login_Success_From** class.
- The **extra** attribute was defined as a temporary attribute. It is not forwarded to the event server as a part of this event.

There are a couple of other interesting items to note from this example:

- In the **PRINTF** value specification for the **msg** attribute in the **Root_Login_Success_From** class, two **%s** conversions are specified without any intervening white space. This allows the final **msg** attribute value to be created without any space between the string **oak** and the comma.
- In the **Root_Login** format specification, there are no explicit mappings; all mappings are inherited. This allows class name specialization without changing any attribute values. Any event that matches the **Logfile_Login** class has the same attributes and values as those that match the **Root_Login** class, but the class name is different.
- Variables are resolved from the matching format specification, even if they are inherited. For example, if the **msg** attribute had not been overridden with the

PRINTF statement in the **Root_Login_Success_From** class, its value would have been **ttyp6**. This is because the **msg** attribute is inherited as the third component specification in the event, even though the third component in the originating class (**Logfile_Base**) would have yielded the value **sawmill login: ROOT LOGIN ttyp6 FROM oak**.

Activating Changes Made with a Format File

If you have made changes to a format file, you must generate a new class definition statement (CDS) file that contains those changes.

Generating a New Class Definition Statement File for a TME Adapter

To generate a new CDS file for a TME adapter, simply distribute a profile containing the changed format file to the appropriate endpoints. The shipped default profile contains the appropriate commands to automatically perform the following actions:

1. Stop the adapter.
2. Generate a new CDS file from the distributed format file.
3. Restart the adapter.

These commands can be viewed for the profile being distributed by selecting **Actions** in the **Edit Adapter** window of the ACF.

Generating a New Class Definition Statement File for a Non-TME Adapter

To generate a new CDS file for a non-TME adapter, you must perform the following tasks:

1. Stop the adapter.

   NetWare log file
   See “TECADNW4.NLM” on page 61.

   OS/2 See “Stopping the Adapter” on page 81.

   UNIX log file
   See “Stopping the Adapter” on page 102.

   Windows event log
   See “Stopping the Adapter” on page 120.

   Windows NT event log
   See “Stopping the Adapter” on page 135.

2. Generate a new CDS file using the following commands. The **logfile_gencds, nw4gencds.nlm, os2gncds.exe**, and **nt_gencds.exe** programs are located in the **bin** subdirectory of the directory where you installed the adapter. The format file is in the appropriate language subdirectory in the **etc** directory where you installed the adapter (see “Format File Location” on page 145 for the appropriate language subdirectory). Specify the appropriate path to create the new CDS file in the **etc** directory.

   OS/2

   ```
   os2gncds /language/tecados2.fmt tecados2.cds
   ```

   UNIX log file

   ```
   logfile_gencds /language/tecad_logfile.fmt > tecad_logfile.cds
   ```

Windows event log

```
win_gencds /language/tecad_win.fmt tecad_win.cds
```

Windows NT event log

```
nt_gencds /language/tecad_nt.fmt tecad_nt.cds
```

3. Restart the adapter:

   NetWare log file
   See “TECADNW4.NLM” on page 61.

   OS/2 See “Starting the Adapter” on page 80.

   UNIX log file
   See “Starting the Adapter” on page 101.

   Windows event log
   See “Starting the Adapter” on page 120.

   Windows NT event log
   See “Starting the Adapter” on page 135.

Appendix C. Class Definition Statement File Reference

A class definition statement (CDS) file specifies **SELECT**, **FETCH**, and **MAP** statements for all event classes supported by adapters that utilize a CDS file. This provided file is required for most adapters and has the same format for all adapters that use it. A CDS file has an extension of **.cds**; see each adapter chapter for exact file names.

File Format

Most of the CDS file is composed of class definition statements. A CDS file has the following format:

```
MAP_DEFAULT
  map_default_clause
END
CLASS class_name
SELECT
  select_clause  ...
FETCH
  fetch_clause
  ...
MAP
  map_clause
END
```

Comment lines begin with a number sign (**#**). For syntax reference information in BNF notation, see "Class Definition Statement File Syntax Diagrams" on page 161.

Operators

Various operators are used in class definition statements, as follows:

- The **PREFIX** and **SUFFIX** operators are valid only for string attribute names, values, or keys.
- The **CONTAINS** operator is valid only on string values.
- The not equals (**!=**), greater than (**>**), greater than or equals (**>=**), less than (**<**), and less than or equals (**<=**) operators are applicable only to integer values; they are not implemented for integer keys.

The following is an example of the use of the operators. In this example, the code is for an AS/400 message adapter:

```
CLASS AS400_MSG
SELECT
  1: ATTR(=,$MSG), VALUE(PREFIX,"Job");
  2: ATTR(=,$MSG), VALUE(CONTAINS,"for User");
  3: ATTR(=,$MSG), VALUE(SUFFIX,"You must investigate.");
FETCH
  1: SUBSTR($MSG,4,8)
  2: SUBSTR($MSG,22,8)
MAP
  $severity = CRITICAL;
```

```
  $msg = PRINTF("Job %s for user %s is on message wait", $F1, $F2);
END
```

Table 3 describes each statement in the example:

Table 3. Explanation of operators in example code

Code	Explanation
SELECT `ATTR(=,$MSG), VALUE(`**`PREFIX`**`,"Job");`	A match occurs when any message arriving with the Class=AS400_MSG, where the first part of the message field equals `Job`.
SELECT `ATTR(=,$MSG), VALUE(`**`CONTAINS`**`,"for User");`	A match occurs when the message field contains `for User` anywhere within the message text.
SELECT `ATTR(=,$MSG), VALUE(`**`SUFFIX`**`,"You must investigate.");`	In order to match, the end of the message field must be the text `You must investigate.` The case of the message must be exactly as shown in the example.

FETCH
`SUBSTR($MSG,4,8)`
`SUBSTR($MSG,22,8)`
	This part of the **FETCH** statement pulls characters from the message field. It starts at character 5, because it is zero-based. It pulls a total of eight characters. For example, the message is `Job 12345678 for User stephens has stopped. You must investigate.` The statement pulls `12345678` for the first line of the **FETCH** statement. The second line pulls the text `stephens`.

MAP
`$severity = CRITICAL;`
`$msg = PRINTF("Job %s for user %s is on message wait", $F1, $F2);`
	The severity attribute is set to CRITICAL. It prints using the two items that were pulled with the **FETCH** statement.

Class Definition Statement File Details

For each class of event supported by an adapter, one or more class definition statements are present in the CDS file. These statements define which incoming event maps to a particular class and how the attributes of the formatted event instance going to the event server are filled with values. The class definition statements are described as follows:

SELECT
: Specifies the criteria an incoming event must satisfy to match a class.

FETCH
: Retrieves data from the incoming event that is necessary to fill the attribute values.

MAP
: Specifies how to fill attribute values for an event instance from data retrieved by **FETCH** statements.

Class definition statements are evaluated in the order they appear in the CDS file. An incoming event is mapped to the class specified by the first class definition statement whose **SELECT** statement is evaluated successfully.

When more than one class definition statement is provided for a particular class of event, the class definition statement with the most restrictive **SELECT** statement is placed before the less restrictive statements in the CDS file. Locating the most restrictive class definition statement first for a same-named class provides for better performance of the adapter.

If the class name equals ***DISCARD***, any incoming event matching the **SELECT** statement is discarded. Note that an event is also discarded if it does not match any class definition statement. However, if a particular type of incoming event must always be discarded (for example, routine events that are of no importance to administrators), it is more efficient to define a ***DISCARD*** class definition statement and locate it at the beginning of the CDS file, rather than let the adapter evaluate all class definition statements and finally discard the event.

SELECT Statement

There is one **SELECT** statement for each class definition statement. **SELECT** statements have the following general format, where *n* is the identification number of a clause within a **SELECT** statement:

```
SELECT
n:    ATTR(a_op, a_op_value),
      KEY(k_op, k_op_value),
      VALUE(v_op, v_op_value);
```

The **ATTR** part is mandatory and specifies a condition on the attribute name. The **KEY** and **VALUE** parts are optional and respectively specify a condition on the attribute key and attribute value. *a_op*, *k_op*, and *v_op* are available operators to express conditions over the attribute name, key, or value (=, !=, <, <=, >, >=, PREFIX, SUFFIX, CONTAINS). *a_op_value*, *k_op_value*, and *v_op_value* specify the comparison value.

In order for a **SELECT** statement to be evaluated successfully, the following conditions must be met as follows:

- The incoming event must contain an attribute whose name matches the **ATTR** part. If the match is not unique (that is, several attributes can match the **ATTR** part), only the first match is used. It is the key and value of this attribute that is referred to in the rest of the statement. For example:

  ```
  ATTR(=,"ifDescr")
  ```

  means that the incoming event must contain an attribute named **ifDescr**.
- If a **KEY** part is present, the key of the attribute selected during the previous step must match the condition expressed by the **KEY()** expression. For example:

  ```
  KEY(!=,1)
  ```

  means that attribute **ifDescr** must have a key with a value other than 1.

  Note: AS/400 adapters do not support **KEY** parts in CDS files.
- If a **VALUE** part is present, the value of the attribute must match the condition expressed by the **VALUE** expression. For example:

  ```
  VALUE(PREFIX,"Serial")
  ```

  means that the value of attribute **ifDescr** must begin with **Serial** (for example, **Serial1**).

Using the previous examples, the complete clause of the **SELECT** statement reads as follows:

```
SELECT
1:    ATTR(=,"ifDescr"), KEY(!=,1),
      VALUE(PREFIX,"Serial");
```

SELECT statements and their associated clauses are evaluated in the order they appear in the CDS file. If all the clauses of a **SELECT** statement are evaluated successfully, the incoming event matches the corresponding class.

After an event is matched with a class because of successful **SELECT** statement evaluation, processing continues with the **FETCH** statement, unless the class is ***DISCARD***, in which case the event is discarded. If the evaluation of a **SELECT** statement fails, the kernel tries to match the event with the **SELECT** statement of the next class. If the incoming event cannot be matched with any class, it is discarded.

Each time a **SELECT** statement is evaluated successfully, the adapter kernel layer creates three temporary pseudo-variables: **$N***n*, **$K***n*, **$V***n* (where *n* is the identification number of a clause in the **SELECT** statement). These variables contain the name, key, and value of the attribute specified in the clause, respectively. The pseudo-variables can then be used in any following **SELECT**, **FETCH**, or **MAP** statement.

By default, the attribute name specified in an **ATTR()** expression is a string, and the attribute matching this name is searched for sequentially in the incoming event. For most adapters, every incoming event contains a minimum set of mandatory fields. For this reason, each adapter supports built-in keywords that can be used to reference these mandatory attributes and thereby directly access their values. These keywords have the format **$***attribute_name*. Examples of keywords supported by the SNMP adapter are: **$AGENT_ADDRESS**, **$COMMUNITY**, **$ENTERPRISE**, **$TYPE**, and **$SPECIFIC**. These keywords refer to the mandatory fields of an SNMP Trap-PDU. Each adapter can also define global variables, such as **RECEPTION TIME**, **SVARBIND**, and so forth.

Using the **$** notation, a clause for SNMP authentication failure traps can be written as follows:

```
1:  ATTR(=,$TYPE),VALUE(=,4);
```

This notation is not simpler than the format shown in the previous example, ATTR(=, ″type″), but evaluation will be faster since it results in direct access to the variable instead of a linear search.

The syntax shown in the preceding example is generic, and as such, it can be rather verbose for commonly used criteria. Several shortcuts are provided in order to alleviate the notation. For example, the previous example can be written as follows:

```
1:$TYPE=4;
```

Output from the class selection process is the name of the event class, a table of pseudo-variables **$N***n*, **$K***n*, **$**V*n*, and all adapter-specific variables (for example, **$TYPE**, **$VARBIND**, and so forth).

FETCH Statement

The **FETCH** statement of a class definition statement allows manipulation and modification to the attribute names, keys, and values retrieved by the **SELECT** statement for the incoming event. Sometimes it is necessary to perform tasks such as extracting a substring from an attribute value, adding two values, and so forth.

There can be one or more clauses within a **FETCH** statement. Each clause has the following format:

n:*expression*;

where *n* is the identification number of a clause within a **FETCH** statement and *expression* is an expression specifying the value to assign the pseudo-variable **$F***n*. Pseudo-variables are the output from a clause of a **FETCH** statement. This expression can make reference to any pseudo-variable defined by the adapter, which could have been created from the **SELECT** statement or from a previous clause within the **FETCH** statement for the class.

An example of a **FETCH** statement is the following:

```
FETCH
  1:SUBSTR ($V2, 1, 5 );
```

MAP Statement

The MAP statement of a class definition statement assigns values to the attributes of the event class instance.

There can be one or more clauses in a MAP statement. Each clause has one of the following two formats:

attribute_name=variable;

attribute_name=PRINTF(*format_string*,*var1*,...);

An example of a MAP statement is the following:

```
MAP
  origin=$AGENT-ADDRESS;
  msg=PRINTF("Link %s is DOWN",$V3);
```

The output from a **MAP** statement is a list of attribute *name=value* pairs that is used to generate the outgoing event for the event server.

MAP_DEFAULT Statement

Some attributes, like **source** and **sub_source,** could have a constant value for all the events generated by an adapter type. To not repeat identical clauses for **MAP** statements in all class definition statements for an adapter, the CDS file can contain a **MAP_DEFAULT** statement. The **MAP_DEFAULT** statement specifies default values for the mandatory attribute *name=value* pairs. The following example illustrates a **MAP_DEFAULT** statement:

```
MAP_DEFAULT
  source = SNMP;
  sub_source = NET;
# forwarding_agent = $SOURCE_ADDR;
  origin = $AGENT_ADDR;
  adapter_host = $ADAPTER_HOST;
END
```

Example

The following example shows a CDS file:

```
#
# Default attribute values
#
MAP_DEFAULT
```

```
  source=NET;
  sub_source=SNMP-TRAP;
  origin=$SOURCE_ADDR;
END

CLASS Authentication_Failure_Cisco
  SELECT
    1: ATTR(=,$ENTERPRISE), VALUE(PREFIX, "1.3.6.1.4.1.9");
    2: $TYPE = 4;
    3: ATTR(=,"authAddr");
  FETCH
    1: IPNAME($SOURCE_ADDR);
  MAP
    hostname = $F1;
    originating_address = $V3;
END
# For Cisco routers, because we know the interface generating the trap,
# we map 'linkUp' traps to 'linkDown' CLOSED events
CLASS Link_Down_Cisco
  SELECT
    1: ATTR(=,$ENTERPRISE), VALUE(PREFIX, "1.3.6.1.4.1.9");
    2: $TYPE = 3;
    3: ATTR(=,"ifIndex");
    4: ATTR(=,"ifDescr");
    5: ATTR(=,"ifType");
    6: ATTR(=,"locIfReason");
  FETCH
    1: IPNAME($SOURCE_ADDR);
  MAP
    hostname = $F1;
    sub_origin = $V4;
    status = CLOSED;
    interface_index = $V3;
    interface_description = $V4;
    interface_type = $V5;
    reason = $V6;
END
```

Object Identifier to Name Translation

The selection of an attribute is based on its name. With adapters that receive SNMP trap messages, the standard way of naming attributes is to use object identifiers (OIDs). For example, SNMP variable **ifDescr** is named **1.3.6.1.2.1.2.2.1.2**. Using SNMP object identifiers in SELECT statements is not very convenient. Additionally, since the SNMP variable **ifDescr** is part of a table, it will be indexed by the interface number. If the interface number is **2**, the received object identifier will be **1.3.6.1.2.1.2.2.1.2.2**. Without some knowledge of the Management Information Base (MIB), the SNMP adapter has no way to translate an object identifier into a more understandable name, or to extract key parts from an object identifier.

An object identifier file (**tecad_***adaptername***.oid**) for SNMP-based adapters contains OID-to-name mappings for some SNMP variables. You can add or modify this file as needed. The format of an object identifier file is:

name object_identifier

For example:

```
"authAddr"    "1.3.6.1.4.1.9.2.1.5"
"ifDescr"     "1.3.6.1.2.1.2.2.1.2"
```

Class Definition Statement File Syntax Diagrams

This section describes the syntax for statements allowed within a CDS file. The syntax is shown in BNF-like notation where the vertical bar (|) character represents alternatives, and optional parts are contained within braces ({}).

```
*
 *  FILE CONTENT
 */
<file> ::= <statements> | /* empty */
<statements> ::=
    <statement>
  |  <statement> <statements>
<statement> ::=
    <mapdefault_statement>
  |  <class_statement>
/*
 *  MAP_DEFAULT STATEMENT
 */
<mapdefault_statement> ::=
    MAP DEFAULT
      <mapdef_statements>
    END
<mapdef_statements> ::=
    <mapdef_statement>
  |  <mapdef_statement> <mapdef_statements>
<mapdef_Statement> ::=
    <attribute_name>  '=' <constant> ';'
  |  <attribute_name>  '=' <keyword> ';'
<attribute_name> ::+ <atom>
/*
 *  CLASS STATEMENT
 */
<class_statement> ::=
    CLASS <class_name>
  {  SELECT
      <select_statements>                          }
  {  FETCH
      <fetch_statements>                          }
{    MAP
      <map_statements>                           }
    END
<class_name> ::+
    *DISCARD*
  |  <atom>
/*
 *  SELECT STATEMENT
 */
<select_statements> ::=
    <select_statement>
  |  <select_statement> <select_statements>
<select_statement> ::=
    <number>      ':' <attr_decl>
      {    ',' <key_decl>                       }
      {    ',' <value_decl>                       }
          ';'
  |  <number>      ':' <keyword> '=' <v_op_val> ';"
  |  <number>      ':' <constant> '=' <v_op_val> ';'
<attr_decl> ::=
    ATTR '(' <a_op> ',' <a_op_val> ')'
```

```
<key_decl ::=
    KEY '(' <k_op> ',' <v_op_val> ')'
<a_op> ::=
    '='
  |  PREFIX
  |  SUFFIX
  |  EXISTS


<a_op_val> ::=
    <constant>
  |  <keyword>
  |  <name_var>
  |  <key_var>
  |  <value_var>
<k_op> ::=
    '='
  |  '!='
  |  '>'
  |  '>='
  |  '<'
  |  '<='
  |  PREFIX
  |  SUFFIX
  |  EXISTS
<k_op_val> ::=
    <constant>
  |  <keyword>
  |  <name_var>
  |  <key_var>
  |  <value_var>
<v_op> ::=
    '='
  |  '!='
  |  '>'
  |  '>='
  |  '<'
  |  '<='
  |  PREFIX
  |  SUFFIX
  |  EXISTS
<v_op_val> ::=
    <constant>
  |  <keyword>
  |  <name_var>
  |  <key_var>
  |  <value_var>
/*
 *  FETCH STATEMENT
 */
<fetch statements> ::=
    <fetch_statement>
  |  <fetch_statement> <fetch_statements>
<fetch_statement> ::= <number> ':' <fetch_expr> ';'
<fetch_expr> ::=
    <fetch_value>
  |  <substr_expr>
<fetch value> ::=
    <constant>
  |  <keyword>
```

```
  |  <name_var>
  |  ckey_var>
  |  <value_var>
  |  <fetch_var>
<substr_expr> ::=
    SUBSTR '(' <fetch_expr> '.'
           <fetch_expr> ','
           <fetch_expr> ')'
/*
 *  MAP STATEMENT
 */
<map_statements> ::=
    <map_statement>
  |  <map_statement> <map_statements>
<map_statement> ::=
    <attribute_name> '=' <map value> ';'
  |  <attribute_name> '=' PRINTF '(' <string> ','
    <map_args> ')' ';'
<map_args> ::=
    <map_value>
  |  <map_value> ',' <map args>
<map value> ::=
    <constant>
  |  <keyword>
  |  <name_var>
  |  <value_var>
  |  <fetch_var>
/*
 *  VARIOUS
 */
<constant> ::=
    <string>              e.g. hello, "hello"
    <number>                  12
<keyword> ::= '$<atom>'                 e.g. $TARGET
<name_var> ::= '$N<number>'                 e.g. $N12
<key_var> ::= '$K<number>'                 e.g. $K12
<value_var> ::= '$V<number>'                 e.g. $V5
<fetch_var> ::= '$F<number>'                 e.g. $F2
<string> ::=-
    <quoted_string>
  |  <atom>
<quoted_string> ::=                 e.g. "sun", "a ""dog"" !"
<atom> ::=            e.g. target, C3000, LINKD_DOWN, in-out
```

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents.You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation

Licensing

2-31 Roppongi 3-chome, Minato-ku

Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION ″AS IS″ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

2Z4A/101

11400 Burnet Road

Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to

IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

If you are viewing this information in softcopy form, the photographs and color illustrations might not appear.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX	IBM Logo	Tivoli
AS/400	Integrated Language Environment	Tivoli Logo
FFST	NetView	Tivoli Enterprise
First Failure Support Technology	OS/2	Tivoli Enterprise Console
IBM	OS/390	TME

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Glossary

The following cross-references are used in this glossary:

See: This refers the reader to (a) a related term, (b) a term that is the expanded form of an abbreviation or acronym, or (c) a synonym or more preferred term.

Obsolete term for:
This indicates that the term should not be used and refers the reader to the preferred term.

A

ACF. See Adapter Configuration Facility.

ACP. See adapter configuration profile.

adapter. See event adapter.

Adapter Configuration Facility (ACF). In the IBM Tivoli Enterprise Console product, a graphical user interface that enables a Tivoli administrator to easily configure and customize event adapters.

adapter configuration profile (ACP). In a Tivoli environment, an IBM Tivoli Enterprise Console profile that contains information for one or more event adapters.

attribute. A characteristic that identifies and describes a managed object. The characteristic can be determined, and possibly changed, through operations on the managed object.

authorization role. In a Tivoli environment, a role assigned to Tivoli administrators to enable them to perform their assigned systems management tasks. A role may be granted over the entire Tivoli management region or over a specific set of resources, such as those contained in a policy region. Examples of authorization roles include super, senior, admin, and user.

B

BAROC. See Basic Recorder of Objects in C.

Basic Recorder of Objects in C (BAROC). In the event server of the IBM Tivoli Enterprise Console product, the internal representation of the defined event classes.

C

CDS. See class definition statement.

class definition statement (CDS). For the IBM Tivoli Enterprise Console product, a statement that specifies (a) the mapping of incoming events to classes and (b) the values assigned to event attributes.

configuration file. A file that specifies the characteristics of a system device or network.

E

endpoint. (1) In a Tivoli environment, a Tivoli client that is the ultimate recipient for any type of Tivoli operation. (2) In a Tivoli environment, a Tivoli service that runs on multiple operating systems and performs Tivoli operations on those systems, thereby enabling the Tivoli Management Framework to manage the systems as Tivoli clients.

event. In the Tivoli environment, any significant change in the state of a system resource, network resource, or network application. An event can be generated for a problem, for the resolution of a problem, or for the successful completion of a task. Examples of events are: the normal starting and stopping of a process, the abnormal termination of a process, and the malfunctioning of a server.

event adapter. In a Tivoli environment, software that converts events into a format that the IBM Tivoli Enterprise Console product can use and forwards the events to the event server. Using the Tivoli Event Integration Facility, an organization can develop its own event adapters, tailored to its network environment and specific needs.

event class. In the IBM Tivoli Enterprise Console product, a classification for an event that indicates the type of information that the event adapter will send to the event server.

event console. In the IBM Tivoli Enterprise Console product, a graphical user interface (GUI) that enables system administrators to view and respond to dispatched events from the event server. The Tivoli Event Integration Facility does not directly use or affect event consoles.

event group. In the IBM Tivoli Enterprise Console product, a set of events that meet certain criteria. Each event group is represented by an icon on the event console. Tivoli administrators can monitor event groups that are relevant to their specific areas of responsibility.

event server. In the IBM Tivoli Enterprise Console product, a central server that processes events. The event server creates an entry for each incoming event and evaluates the event against a rule base to determine whether it can respond to or modify the event automatically. The event server also updates the event consoles with the current event information. If the primary event server is not available, events can be sent to a secondary event server.

F

format file. A file that serves as the lookup file for matching log messages to event classes. Moreover, it serves as the source from which a CDS file is generated.

G

gateway. See IBM Tivoli Enterprise Console gateway and Tivoli Management Framework gateway.

I

IBM Tivoli Enterprise Console gateway. The software that receives events from endpoint adapters, bundles them up, and forwards them to the event server. It runs on the same host as the Tivoli Management Framework gateway.

M

managed node. In a Tivoli environment, a computer system on which Tivoli Management Framework is installed. Contrast with endpoint.

P

prefilter. A type of filter defined in an adapter configuration file for filtering raw events at the source, before any adapter processing.

profile. In a Tivoli environment, a container for application-specific information about a particular type of resource. A Tivoli application specifies the template for its profiles; the template includes information about the resources that can be managed by that Tivoli application.

A profile is created in the context of a profile manager; the profile manager links a profile to the Tivoli resource (for example, a managed node) that uses the information contained in the profile. A profile does not have any direct subscribers.

R

rule. In the IBM Tivoli Enterprise Console product, one or more logical statements that enable the event server to recognize relationships among events (event correlation) and to execute automated responses accordingly. Also see rule base and rule set.

rule base. In the IBM Tivoli Enterprise Console product, one or more rule sets and the event class definitions for which the rules are written. The IBM Tivoli Enterprise Console product uses the rule base in managing events. An organization can create many rule bases, with each rule base fulfilling a different set of needs for network computing management.

rule set. In the IBM Tivoli Enterprise Console product, a file that contains one or more rules. Also see rule base.

S

severity level. In the IBM Tivoli Enterprise Console product, a classification for an event that indicates its degree of severity. Severity levels can be modified by a user or an IBM Tivoli Enterprise Console rule. The predefined severity levels, in order of descending severity, include: fatal, critical, warning, minor, harmless, and unknown.

slot. In the IBM Tivoli Enterprise Console product, obsolete term for attribute.

source. In the IBM Tivoli Enterprise Console product, a resource, such as a host, that is being monitored by an event adapter.

T

task. In a Tivoli environment, the definition of an action that must be routinely performed on various managed nodes throughout the network. A task defines the executable files to be run when the task is executed, the authorization role required to execute the task, and the user or group name under which the task will execute.

Tivoli Management Framework gateway. Tivoli Management Framework software that provides services between the endpoints and the rest of the Tivoli environment. Also referred to as the endpoint gateway.

Tivoli management region (TMR). In a Tivoli environment, a Tivoli server and the set of clients that it serves. An organization can have more than one TMR. A TMR addresses the physical connectivity of resources whereas a policy region addresses the logical organization of resources.

TMR. See Tivoli Management Region.

Index

Special characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Program Number: 5698-TEC

Printed in U.S.A.

GC32-0668-01